Correlation Power Analysis of Lightweight Block Ciphers: From Theory to Practice

Alex Biryukov, Daniel Dinu, and Johann Großschädl

SnT, University of Luxembourg
6, rue Richard Coudenhove-Kalergi, L–1359 Luxembourg, Luxembourg
`{alex.biryukov,dumitru-daniel.dinu,johann.groszschaedl}@uni.lu`

Abstract. Side-Channel Analysis (SCA) represents a serious threat to the security of millions of smart devices that form part of the so-called Internet of Things (IoT). Choosing the "right" cryptographic primitive for the IoT is a highly challenging task due to the resource constraints of IoT devices and the variety of primitives. An important criterion to assess the suitability of a lightweight cipher with respect to SCA is the amount of leakage available to an adversary. In this paper, we analyze the efficiency of different selection functions that are commonly used in Correlation Power Analysis (CPA) attacks on symmetric primitives. To this end, we attacked implementations of the lightweight block ciphers AES, Fantomas, LBlock, Piccolo, PRINCE, RC5, Simon, and Speck on an 8-bit AVR processor. By exploring the relation between the nonlinearity of the studied selection functions and the measured leakages, we discovered some imperfections when using nonlinearity to quantify the resilience against CPA. Then, we applied these findings in an evaluation of the "intrinsic" CPA-resistance of unprotected implementations of the eight mentioned ciphers. We show that certain implementation aspects can influence the leakage level and try to explain why. Our results shed new light on the resilience of basic operations executed by these ciphers against CPA and help to bridge the gap between theory and practice.

Keywords: CPA, Selection function, Leakage, Nonlinearity

1 Introduction

Side-Channel Analysis (SCA) [21] belongs to the genre of physical attacks and exploits some auxiliary information (e.g. the power consumption leaking from a device that executes a cryptographic algorithm) to recover the secret key. The history of SCA stretches back 20 years when Kocher described the first timing attacks [24] and thereafter introduced the basics of Differential Power Analysis (DPA) [23]. Since then, non-invasive attacks exploiting the power consumption or electromagnetic emanations of a target device have been steadily improved by using better leakage models and advanced analysis techniques to recover the secret key. Notable milestones in the evolution of power analysis attacks in the past 15 years include Correlation Power Analysis (CPA) [7], Template Attacks (TA) [10], and Mutual Information Analysis (MIA) [16].

The study of "lightweight" symmetric primitives has been a hot topic in the cryptographic community in the past few years, driven primarily by the rapid growth of the Internet of Things (IoT) [14] and the demand for security at low cost in terms of execution time, power consumption, RAM requirements, and code size. A significant portion of the smart devices that will soon populate the IoT in large quantities are equipped with an 8-bit microcontroller and feature only a few kB of RAM (e.g. wireless sensor nodes). Such resource constraints pose a massive challenge for the implementation of measures to minimize side-channel leakage, which makes IoT devices an easy target for attacks [22]. It is widely believed in the cryptographic community that side-channel attacks are primarily an implementation problem rather than a design problem, i.e. there is little that can be done from a designer's perspective to eliminate or reduce the leakage of sensitive information. However, some recent research results start to challenge this view, and so does the present work.

Previous research at the intersection between lightweight cryptography and SCA focussed (almost) exclusively on the AES, i.e. there exist only few papers that deal with attacks or countermeasures for other ciphers. In particular, the study of the SCA-resistance of software implementations of lightweight ciphers did not keep pace with the high number of new proposals. In [1], the resilience of the AES and three lightweight block ciphers that share some characteristics (namely KLEIN, LED, and PRESENT) is investigated against profiled single-trace attacks. Unprotected hardware implementations of Simon and LED were analyzed with respect to DPA in [34]. An evaluation of both an unmasked and a masked implementation of Simon for FPGAs was reported in [5]. In [33], the vulnerability of PRINCE and RECTANGLE against DPA is studied. A second line of research focussed on the design of new ligthweight primitives that can be efficiently protected against DPA via masking; representative examples include PICARO [30], Zorro [15], and the LS-designs Robin and Fantomas [17].

The above-mentioned studies on DPA attacks against (lightweight) ciphers other than the AES were mainly "isolated" efforts in the sense that they were carried out on different execution platforms with different measurement setups and different analysis frameworks. A comparative (and consistent) study of the DPA-vulnerability of lightweight block ciphers based on power traces acquired with the same target device is, to our knowledge, still missing. However, such a study would allow one to answer the question of whether different ciphers are equally difficult to attack or not (and if not, why not). Furthermore, we could not find a detailed analysis of the power leakage of basic operations (e.g. arithmetic and logical computations, table look-ups) executed in the round function of common lightweight ciphers. Thus, in this paper, we first try to answer the following questions: (1) How do the theoretical metrics used to assess leakage relate to real-world attack results? (2) Which operation leaks more? Then, we apply the answers of these questions to illustrate how eight lightweight ciphers (namely AES, Fantomas, LBlock [37], Piccolo [35], PRINCE [6], RC5 [32], as well as Simon and Speck [2]) behave with respect to CPA. These eight ciphers were selected from the portfolio of lightweight symmetric algorithms evaluated in [13] using the FELICS framework [11]. The two main selection criteria were high performance and to have a variety of different design strategies.

All results and findings we describe in this paper are based on CPA attacks performed with power consumption traces that were captured on an evaluation board equipped with an 8-bit AVR microcontroller. Our choice for this specific platform is motivated by the widespread use of the 8-bit AVR architecture in resource-limited environments and its particular relevance in the context of the IoT (e.g. wireless sensor nodes). A better understanding of the actual leakage of different operations on 8-bit AVR microcontrollers could influence the design of new lightweight ciphers for the IoT and the implementation of more effective and less costly SCA countermeasures. For example, it is a known fact that the AES leaks significantly due to its highly nonlinear S-box [8], but modern lightweight ciphers generally use smaller S-boxes with lower nonlinearity compared to the AES, and thus one might expect that they leak less. However, an actual confirmation of this assumption with measured traces is still lacking.

We remark that the evaluation of candidates for the NIST SHA-3 standard considered besides security and performance on various hardware and software platforms also SCA resistance as a selection criterion (see e.g. [4, 39] for some concrete results). Currently, a number of standardization bodies, including the NIST, are either considering or have already started the process to standardize lightweight symmetric primitives for the IoT. In this context, it makes sense to compare different aspects of potential candidates, including the SCA resistance of (unprotected) software implementations, before deploying them on millions or even billions of devices. Furthermore, we hope that our work will contribute to a better understanding of how to design lightweight block ciphers that have a better *intrinsic* resistance against side-channel attacks.

Research Contributions. Firstly, we quantify the leakage generated by the execution of different instructions on an AVR processor, aiming to identify the instructions that leak most. Then, we compare the power consumption leakage of basic operations widely used by lightweight ciphers. For each operation, we analyze the relation between our experimental results, the nonlinearity of the operation, and the size (in bits) of the attacked intermediate value.

Secondly, we provide a fair comparison of the resilience of eight lightweight block ciphers against CPA attacks. Knowing which instructions and operations leak more, and knowing all implementation details of the eight ciphers helps to identify the weakest point of each cipher, which can be attacked with maximal efficiency. Our experimental results show that, in some cases, the actual leakage is lower than expected due to certain implementation-related aspects.

The practical approach we follow has the benefit that it gives more realistic results compared with simulated power traces, where the noise is modeled in a deterministic way, which favors the attacker. Thus, our work sheds new light on the resilience of different operations against CPA attacks, and we illustrate this for a set of eight lightweight block ciphers. To the best of our knowledge, there has been no similar effort published in the literature.

2 Preliminaries

Unless specified otherwise, we will use the notation defined in this section. We use the following operators for the corresponding (bitwise) logical operations: "$\cdot$" for AND, "$+$" for OR, "$\oplus$" for XOR. The operators "$\boxplus$" and "$\boxminus$" denote a modular addition and a modular subtraction, respectively. The two functions $\mathsf{MSB}(x)$ and $\mathsf{LSB}(x)$ are used to extract the most and the least significant byte from a stream of bits x, respectively. We represent the S-box layer of a block cipher α by S_α, which may involve the application of one or more S-boxes in parallel, depending on the input size and the specifications of the cipher. The symbol $L^{-1}_{i,Fantomas}$ stands for the result of the inverse linear layer of Fantomas computed with L-box i, where $i \in \{0,1\}$. Finally, $\mathsf{HW}(x)$ denotes the Hamming weight of x, whereas $\mathsf{HD}(x,y) = \mathsf{HW}(x \oplus y)$ is the Hamming distance between x and y.

Definition 1 (Iterated Block Cipher). *An iterated block cipher, sometimes called a product cipher, is a block cipher obtained by iterating r times a round function $R : \{0,1\}^n \to \{0,1\}^n$, each time with its own key $K_i \in \mathcal{K}$, where $\mathcal{K}$ is called round key space. The cipher block size is n bits, the number of rounds is equal to r, $X^{(0)}$ is the plaintext, and $X^{(r)}$ is the ciphertext. It works as follows:*

$$X^{(i)} = R_{K_i}(X^{(i-1)}) \text{ for } 1 \leq i \leq r$$

Definition 2 (Selection Function). *In the context of side-channel attacks, a selection function gives the intermediate result, also referred to as sensitive value $\phi_k = \varphi(x,k)$, which is used by the attacker to recover the secret key. It depends on a known part x of the input $X^{(i-1)}$ of the round function R_{K_i} and on an unknown part k of the round key K_i.*

The attacker computes the intermediate values ϕ_k for a fixed (either known or chosen) input x and for all possible subkeys k. The bit-size $|k|$ of the subkey k determines the memory complexity m of the side-channel attack. Then, she uses the sensitive values $\phi_1, \phi_2, \ldots, \phi_{2^{|k|}}$ and the side-channel leakage to guess the subkey k^* used during the actual computations on the target device. The higher the number of inputs x for which the attacker manages to measure the leakage, the higher the chances to recover the subkey k^*. Usually, the selection functions are chosen to be easy to compute, typically at the first round of the encryption or decryption operation.

Definition 3 (Correlation Power Analysis (CPA)). *Given a set of power traces and the corresponding sets of intermediate values $\phi_1, \phi_2, \ldots \phi_{2^{|k|}}$, Correlation Power Analysis (CPA) aims at recovering the secret subkey k^* using a correlation factor between the measured power samples and the power model of the computed sensitive values.*

The concept of CPA was studied as an improvement of DPA and formalized in [7]. A power model is used to describe the hypothetical power consumption of the target device as a function of the intermediate value ϕ_k considering the device's power consumption characteristics. The Hamming weight (HW) model is more common for software implementations, whereas the Hamming distance (HD) model is generally used for hardware devices.

2.1 Theoretical Metrics for the SCA Resistance of S-Boxes

In the definitions introduced in this subsection, we denote by "+" the addition of integers in $\mathbb{Z}$ and by "$\oplus$" the addition mod 2. We will also use "+" for the addition of two vectors in $\mathbb{F}_2^n$ since there is no ambiguity. For a pair of vectors $a = (a_1, a_2, \ldots, a_m)$ and $b = (b_1, b_2, \ldots, b_m)$ from $\mathbb{F}_2^m$, the scalar product $a \cdot b$ is defined as $a \cdot b = \oplus_{i=1}^m a_i \cdot b_i$.

One way to achieve nonlinearity in symmetric cryptographic primitives is to use S-boxes. Formally, an S-box is an (n, m) function $F : \mathbb{F}_2^n \mapsto \mathbb{F}_2^m$ that maps n input bits to m output bits. If $m = 1$, then F is nothing else than a Boolean function. For any given (n, m) function F, we denote by $(F_1, F_2, \ldots, F_m)$ the coordinate functions of F, such that $F(x) = (F_1(x), F_2(x), \ldots, F_m(x))$, where $F_i : \mathbb{F}_2^n \mapsto \mathbb{F}_2$ for $1 \leq i \leq m$. The *derivative* of F with respect to a vector a in $\mathbb{F}_2^n$ is the function $D_aF : \mathbb{F}_2^n \mapsto \mathbb{F}_2^m$ such that $D_aF(x) = F(x) + F(x + a)$. The *Walsh transform* of F is the function $W_F(u, v) = \sum_{x \in \mathbb{F}_2^n} (-1)^{v \cdot F(x) + u \cdot x}$, while the *cross-correlation transform* of Boolean functions F_i and F_j with respect to a vector $a \in \mathbb{F}_2^n$ is defined as $C_{F_i,F_j}(a) = \sum_{x \in \mathbb{F}_2^n} (-1)^{F_i(x)+F_j(x+a)}$.

Definition 4 (Nonlinearity). *The nonlinearity of an (n, m) function F is defined as:*

$$\mathsf{NL}(F) = 2^{n-1} - \frac{1}{2} \max_{\substack{u \in \mathbb{F}_2^n \\ v \in F_2^{m*}}} |W_F(u, v)| \tag{1}$$

Nonlinearity characterizes the resistance of F against linear cryptanalysis [27]. The higher the nonlinearity of a function, the more resistant the function is to linear cryptanalysis. It is widely accepted that the higher the nonlinearity of a function F, the more information it leaks through side channels.

Definition 5 (Transparency Order). *The Transparency Order of an (n, m) function F, where n and m are two positive integers, is:*

$$\mathsf{TO}(F) = \max_{\beta \in \mathbb{F}_2^m} \left(\left| m - 2\mathsf{HW}(\beta) \right| - \frac{1}{2^{2n} - 2^n} \sum_{a \in \mathbb{F}_2^{n*}} \left| \sum_{\substack{v \in \mathbb{F}_2^m \\ \mathsf{H}(v)=1}} (-1)^{v \cdot \beta} W_{DaF}(0, v) \right| \right)$$

The *Transparency Order* was introduced in [31] to "quantify" the resistance of an S-box against DPA attacks using the Hamming weight power model. In general, the smaller the transparency order of F, the higher is its resistance to DPA attacks. $\mathsf{TO}(F)$ satisfies the following relation: $0 \leq \mathsf{TO}(F) \leq m$.

Definition 6 (Improved Transparency Order). *The Improved Transparency Order of a balanced (n, m) function F is defined as:*

$$\mathsf{ITO}(F) = \max_{\beta \in \mathbb{F}_2^m} \left(m - \frac{1}{2^{2n} - 2^n} \sum_{a \in \mathbb{F}_2^{n*}} \sum_{j=1}^{m} \left| \sum_{i=1}^{m} (-1)^{\beta_i + \beta_j} C_{F_i, F_j}(a) \right| \right)$$

The *Improved Transparency Order* addresses the limitations identified in the initial definition of TO [9].

Definition 7 (DPA Signal-to-Noise Ratio). *The DPA Signal-to-Noise Ratio of function F is defined as:*

$$\mathsf{SNR}(F) = m 2^{2n} \left(\sum_{a \in \mathbb{F}_2^n} \left(\sum_{i=0}^{m-1} \left(\sum_{x \in \mathbb{F}_2^n} (-1)^{F_i(x) + x \cdot a} \right) \right)^4 \right)^{-\frac{1}{2}}$$

The *DPA Signal-to-Noise Ratio* was proposed in [18] as a way to model the information leakage of CMOS circuits using the tools of traditional cryptanalysis. The SNR increases when the resistance of an S-box to linear and differential cryptanalysis increases. A novel definition of the SNR based on the maximum likelihood estimator was introduced in [19].

3 Evaluation Framework

Measurement Setup. All experiments reported in this paper were performed on an evaluation board equipped with an 8-bit ATmega2561 processor clocked at 16 MHz. A regulated power supply provides the 5 V supply voltage required for the operation of the board. The evaluation board and the computer used to control the measurements are connected through optical fiber. We placed the board in a Faraday cage to reduce the environmental noise. The measurements of the power traces were performed with a LeCroy waveRunner 104MXi digital sampling oscilloscope using a differential probe.

We mounted the CPA attacks against the ANSI C implementations of the selected ciphers available in the FELICS framework [11]. The only modification of the original C source codes we made was the insertion of a trigger signal to indicate the beginning and the end of the side-channel relevant portion of the power traces. To have a common ground for comparison, we assumed that the attacker needs to recover the 32 bits of the round key $K_1 = \text{0x01234567}$ for all eight block ciphers. Note that, in all of our experiments, we acquired the same number of traces, namely q for the encryption of q known plaintexts.

Metrics. To ensure a fair and uniform side-channel evaluation of the selected ciphers, we used the evaluation methodology for key-recovery attacks proposed in [36]. In that paper, two different types of evaluation metrics are defined: an information-theoretic metric quantifying the amount of information that leaks from a given implementation, and an actual security metric, which quantifies how well the leaked information can be used by the attacker.

Since we conducted a practical evaluation based on leakages acquired from a target board using the described setup instead of attacks based on simulated power traces, the actual security metrics (i.e. success rate and guessing entropy) are sound for our study. We do not use the information-theoretic metric from [36] (i.e. conditional entropy) because it involves profiling the target device in order to approximate the probability distribution of the leakage, which reduces the applicability of the attack to a certain class of devices. Moreover, both the template creation and the approximation of the probability distribution for all leakage samples are computationally intensive.

We recall that side-channel attacks are generally performed using a divide-and-conquer approach. The adversary attacks a subkey class κ with $|\kappa| \ll |\mathcal{K}|$ using the selection function $\varphi(x,k)$ and q measurements. As result she gets a guess vector $g = [g_1, g_2, \ldots, g_{2^{|k|}}]$ for the subkey k with the possible candidates sorted in descending order, the most-likely subkey candidate being g_1, and the least-likely subkey candidate being $g_{2^{|k|}}$. The following two metrics quantify the amount of effort required to recover the correct subkey k^* from the guess vector. Consequently, they serve as an indicator of how efficient an attack is in the case of q measurement queries.

Definition 8 (Success Rate). *The success rate of order o, $o \leq 2^{|k|}$, of a side-channel key recovery attack is defined as:*

$$\mathsf{SR}_o(k^*, g) = \begin{cases} 1, & \textit{if } k^* \in [g_1, g_2, \ldots, g_o] \\ 0, & \textit{otherwise} \end{cases}$$

Definition 9 (Guessing Entropy). *The guessing entropy of a side-channel key recovery attack is:*

$$\mathsf{GE}(k^*, g) = log_2 i, \textit{ such that } k^* = g_i \textit{ for } g_i \in [g_1, g_2, \ldots, g_{2^{|k|}}]$$

Given an implementation $\mathcal{C}$ to be evaluated using N experiments with the maximum number of measurement queries q, the memory complexity m, and the time complexity t, Algorithm 1 shows in detail how the mean success rate of order o, i.e. $\overline{\mathsf{SR}_o^i}$, and the mean guessing entropy, i.e. $\overline{\mathsf{GE}^i}$, can be computed for i power consumption traces. The results are accompanied by the respective standard errors $\mathsf{SE}_{\overline{\mathsf{SR}_o^i}}$ and $\mathsf{SE}_{\overline{\mathsf{GE}^i}}$. Unless otherwise specified, the results in this paper are based on $N = 100$ experiments, each with $q = 2000$ queries. Both the time complexity t and memory complexity m were determined by guesses of at most 8-bit subkeys of the round key K_1, where k^* is the actual key used by the implementation $\mathcal{C}$.

4 Quantifying the Leakage

Using the measurement environment described before, we quantify the leakage of different instructions to find out which instruction gives the “best” target in

Algorithm 1: CPA Evaluation Algorithm

```
Data: $\mathcal{C}$, $k^*$, $q$, $m$, $t$, $N$
Result: $\overline{\mathsf{SR}^i_o}$, $\overline{\mathsf{GE}^i}$, $\mathsf{SE}_{\overline{\mathsf{SR}^i_o}}$, $\mathsf{SE}_{\overline{\mathsf{GE}^i}}$
for $j$ in $N$ do
    $\mathsf{AcquirePowerTraces}(\mathcal{C}, k^*, q)$;
    for $i$ in $q$ do
        $g = \mathsf{CPA}(\mathcal{C}, i, m, t)$;
        compute and store $\mathsf{SR}^{j,i}_o(k^*, g), \mathsf{GE}^{j,i}(k^*, g)$;
    end
end
for $i$ in $q$ do
    compute $\overline{\mathsf{SR}^i_o} = \frac{1}{N}\sum_{j=1}^{N} \mathsf{SR}^{j,i}_o(k^*, g), \overline{\mathsf{GE}^i} = \frac{1}{N}\sum_{j=1}^{N} \mathsf{GE}^{j,i}(k^*, g)$;
    compute $\mathsf{SE}_{\overline{\mathsf{SR}^i_o}}, \mathsf{SE}_{\overline{\mathsf{GE}^i}}$;
end
```

the power traces when performing a CPA attack. For this purpose, we define the *correlation coefficient difference* $\delta = c_{k^*} - c_{k^\diamond}$ as the difference between the correlation coefficient of the correct key k^*, i.e. c_{k^*}, and the correlation coefficient of the most likely key guess $k^\diamond$, i.e. $c_{k^\diamond}$, with $k^\diamond \neq k^*$.

For the measurements we used a simple Assembly code fragment that contains the targeted Assembly instruction guarded by several `nop` instructions to reduce the noise from other operations such as the communication between the board and the computer or the peaks of the trigger signal. The measurements were done with values of the correct key k^* such that $\mathsf{HW}(k^*)$ runs through all possible values once. For a fixed value of the input plaintext x and key k^*, we averaged eight power measurements of the analyzed instruction to get a single power trace. The plaintext took all possible values from 0x00 up to 0xFF; thus the number of traces q is 256. We performed $N = 10$ experiments for each value of k^*.

4.1 Understanding the Device's Leakage

Understanding the device's leakage requires to understand how different Assembly instructions executed by the processor can impact the power consumption of the device. For this purpose, we evaluated two instructions that operate on registers (namely `and` and `add`) as well as three instructions that require access to memory (namely `lpm`, `ld`, and `st`). The `and` instruction performs a bitwise AND of two 8-bit words, while the `add` instruction executes a modular addition of two 8-bit words. Loading an 8-bit word from the Flash memory of the device into a register can be achieved through the `lpm` instruction, whereas loading an 8-bit quantity from RAM into a register requires a `ld` instruction. Finally, the `st` instruction writes the content of an 8-bit register to memory. We used the AES S-box with the index value given by the plaintext XORed with the key to perform the memory accesses.

Table 1. Correlation coefficient difference $\delta = c_{k^*} - c_{k^\diamond}$ between the correlation of the correct key (i.e. c_{k^*}) and the correlation of the most likely key (i.e. $c_{k^\diamond}$) where $k^\diamond \neq k^*$ for different Hamming weights of the correct key k^* ($\bar{\delta}$ and $\mathsf{SE}_{\bar{\delta}}$ are the mean and the standard error for a 95% confidence interval, respectively).

Instr.	Correct key									$\bar{\delta}$	$\mathsf{SE}_{\bar{\delta}}$
	0x00	0x01	0x03	0x07	0x0F	0x1F	0x3F	0x7F	0xFF		
`and`	-0.798	-0.643	-0.577	-0.518	-0.465	-0.392	-0.329	-0.178	-0.016	-0.435	0.183
`add`	0.190	-0.218	-0.160	-0.079	-0.053	0.001	0.049	0.041	0.001	-0.025	0.093
`lpm`	0.376	0.312	0.271	0.219	0.174	0.169	0.164	0.156	0.143	0.220	0.062
`ld`	0.244	0.200	0.178	0.225	0.215	0.226	0.215	0.195	0.222	0.213	0.015
`st`	0.596	0.581	0.578	0.577	0.566	0.594	0.603	0.585	0.592	0.586	0.008

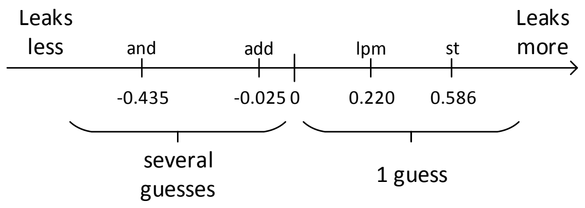

Fig. 1. Correlation coefficient difference spectrum

Our results given in Table 1 show that the memory-access instructions leak a lot more information about the secret key than the register instructions. The writing of a register to memory leaks most, followed by the loading of a word from memory. At the other end of the spectrum is the `and` instruction, which is leaking approximately 20 times less than the `add` instruction (see Table 1 and Fig. 1). We also observed that increasing the number of power traces does not significantly change the values of δ.

Although these experiments may remind the reader about template attacks (where the attacker creates in the profiling phase leakage templates for various instructions), we stress that we did not perform actual template attacks, but we used a technique inspired by classical template attacks to quantify the leakage of different Assembly instructions. Our results indicate that an attacker should target the store of a sensitive value to increase the success rate of the attack.

4.2 Comparison of Different Selection Functions

We now extend the previous experiments to different selection functions, whereby we target the writing of the selection function's result to memory using the `st` instruction, which, as we saw, has the highest leakage. Table 2 summarizes the nonlinearity NL and the mean correlation coefficient difference $\bar{\delta}$ for a total of 16 different selection functions, which are divided into four groups. Detailed values for different correct keys can be found in Table 5.

The first group of selection functions comprises the three logical operations AND, OR, and XOR, which all have a negative value for the mean correlation coefficient difference $\bar{\delta}$. This means that using one of these logical operations as a selection function for a CPA attack is not a very good option. As our results show, only the AND and OR, but not XOR, are sometimes able to recover the correct key k^*, whereby AND is slightly more efficient than OR.

One can notice the contrast between the huge nonlinearity of the AND and OR selection functions on the one side, and all other selection functions listed in Table 2 on the other side. It is also interesting to note that these high values of nonlinearity are accompanied by (relatively) poor values for the correlation coefficient difference. In the case of the bitwise logical operations, it seems the high nonlinearity values do not provide the useful leakage one normally would expect. This contrasts with the conventional wisdom saying that the higher the nonlinearity of a selection function, the more information it leaks in SCA.

Table 2. Leakages of different selection functions (n and m are the input and output size of the selection function in bits, NL is the nonlinearity of the selection function, $\bar{\delta}$ is the mean correlation coefficient difference, and $\mathsf{SE}_{\bar{\delta}}$ is the standard error for a 95% confidence interval).

Selection function	n	m	NL	$\bar{\delta}$	$\mathsf{SE}_{\bar{\delta}}$
$\varphi_1(x,k) = x \cdot k$	16	8	16384	-0.005	0.074
$\varphi_2(x,k) = x + k$	16	8	16384	-0.018	0.060
$\varphi_3(x,k) = x \oplus k$	16	8	0	-0.153	0.168
$\varphi_4(x,k) = x \boxplus k$	16	8	0	0.127	0.011
$\varphi_5(x,k,c) = x \boxplus k \boxplus c$	17	8	0	0.121	0.010
$\varphi_6(x \oplus k) = S_{AES}(x \oplus k)$	8	8	112	0.586	0.008
$\varphi_7(x \oplus k) = S_{LBlock}(x \oplus k)$	4	4	4	0.342	0.008
$\varphi_8(x \oplus k) = S_{LBlock}(x \oplus k)$	8	8	64	0.235	0.006
$\varphi_9(x \oplus k) = S_{Piccolo}(x \oplus k)$	4	4	4	0.339	0.019
$\varphi_{10}(x \oplus k) = S_{Piccolo}(x \oplus k)$	8	8	64	0.259	0.006
$\varphi_{11}(x \oplus k) = S_{PRINCE}(x \oplus k)$	4	4	4	0.269	0.010
$\varphi_{12}(x \oplus k) = S_{PRINCE}(x \oplus k)$	8	8	64	0.138	0.004
$\varphi_{13}(x \oplus k) = \mathsf{LSB}(L^{-1}_{1,Fantomas}(x \oplus k))$	8	8	0	0.087	0.015
$\varphi_{14}(x \oplus k) = \mathsf{MSB}(L^{-1}_{1,Fantomas}(x \oplus k))$	8	8	0	0.041	0.014
$\varphi_{15}(x \oplus k) = \mathsf{LSB}(L^{-1}_{2,Fantomas}(x \oplus k))$	8	8	0	0.136	0.007
$\varphi_{16}(x \oplus k) = \mathsf{MSB}(L^{-1}_{2,Fantomas}(x \oplus k))$	8	8	0	0.083	0.018

The modular addition is similar to the XOR operation; the main difference is the carry propagation in the case of modular addition. Although the nonlinearity of the two modular addition selection functions in Table 2 is zero, there are components of these functions that reach high nonlinearity because of the carry propagation. For clarity, it should be mentioned that all the components of the XOR selection function have a nonlinearity equal to zero, and that the nonlinearity of an (n, m) function is determined by the component having the lowest nonlinearity. By nonlinearity of a component of an (n, m) function F, we mean the nonlinearity of F computed for a fixed vector $v \in \mathbb{F}_2^{m*}$ as shown in Equation (1); see Table 6 for details. This exhibits another imperfection of the nonlinearity metric when used to compare various selection functions regarding side-channel leakage. We note that considering the carry bit c from a previous operation when using selection function φ_5 (`adc` instruction) does not improve the correlation coefficient difference compared with φ_4 (`add` instruction). The modular addition selection function successfully recovered the secret key in all our test cases and should thus be preferred over logical operations.

A further group of selection functions is composed of the substitution layers of different lightweight block ciphers. These selection functions clearly leak the most with respect to CPA. In fact, the selection function using the S-box of the AES has the highest leakage among all studied selection functions. For ciphers using 4-bit S-boxes, we considered two different selection functions: one with an 8-bit input and one with a 4-bit input. The 8-bit selection functions based on the substitution layer of LBlock, Piccolo and PRINCE leak two times less than the selection function using the AES S-box. Surprisingly, although our target device has an 8-bit architecture, the 4-bit selection functions φ_7, φ_9, φ_{11} leak more than the 8-bit selection functions of the same substitution layers.

The selection functions based on the L-boxes of Fantomas are analyzed in a fourth group since they are linear operations, which are generally expected to leak less than nonlinear operations. To our surprise, this group (which consists of the last four selection functions listed in Table 2) leaks more than the logical operations and is on a similar level with the modular addition. Thus, they can be considered as selection functions when performing CPA attacks.

We remark that in [25], the basic algebraic group operations XOR, addition modulo 2^n, and modular multiplication are studied in the context of multi-bit CPA attacks using simulated power traces. Then, selection functions based on the addition modulo 2^{16} and multiplication modulo $2^{16} + 1$ are applied to an implementation of IDEA running on an 8-bit AVR processor. In the case of the modular addition, the characteristics of the correlation coefficients for practical attacks do not correspond to the simulated ones due to signal superposing.

Through these experiments, we revealed some interesting aspects about the leakage of the studied selection functions with respect to CPA. In contradiction to intuitions based on nonlinearity, we made the following observations: 1) the bitwise logical AND and OR operations leak much less than expected and do not always reveal the secret key; 2) for block ciphers that use 4-bit S-boxes, a 4-bit selection function is more efficient than an 8-bit selection function; 3) the linear lookup tables (i.e. L-boxes) used by Fantomas leak more than expected and can be considered as selection functions for CPA attacks.

The lessons we learned from these experiments helped us a lot to select the appropriate leakage functions to attack the eight lightweight block ciphers we briefly describe in the following section.

5 Analyzed Ciphers

We chose the eight lightweight ciphers included in our evaluation according to the following criteria. Firstly, we selected the ciphers from those that achieved good software performance in the Triathlon competition [13]. Besides selecting the ciphers for our CPA study from the ones evaluated in [13], we also used the provided C source codes. This approach has the advantage that all ciphers are implemented according to a common set of guidelines and by the same team of developers, and therefore all implementations had undergone a similar level of optimization. Secondly, we chose our ciphers from the two major structural classes, namely Feistel Networks (FN) and Substitution-Permutation Networks (SPN) with the goal of having many different design approaches with unique features or properties. For example, PRINCE introduced the α-reflection property, which means that a message encrypted under a certain key can only be decrypted with a related key. RC5 introduced data-dependent rotations, while Fantomas is the first instance of the so-called LS-designs.

The main characteristics of the studied ciphers are given in Table 3. In the following, we provide a brief description of each cipher (we refer the reader to the original papers for more details). Half of the eight ciphers use substitution boxes; Table 4 summarizes the most important properties of each S-box.

Table 3. Main characteristics of the analyzed lightweight ciphers.

Cipher	Block Size (bits)	Key Size (bits)	Rounds	Structure	Target Platform	Attacked Operation
AES	128	128	10	SPN	SW, HW	S-box lookup
Fantomas	128	128	12	SPN	SW	L-box lookup
LBlock	64	80	32	Feistel	HW, SW	S-box lookup
Piccolo	64	80	25	Feistel	HW	S-box lookup
PRINCE	64	128	12	SPN	HW	S-box lookup
RC5	64	128	20	Feistel	SW	modular addition
Simon	64	96	42	Feistel	HW, SW	bitwise AND
Speck	64	96	26	Feistel	SW, HW	modular subtraction

AES: Based on the Rijndael block cipher [12], the AES [29] is to date the most important symmetric algorithm. It uses a block size of 128 bits and three different key sizes: 128, 192, and 256 bits. In each round (except for the final round) the `SubBytes`, `ShiftRows`, `MixColumns`, and `AddRoundKey` transformations are applied to a 4×4 byte state matrix. The final round does not include the `MixColumns` transformation. The key schedule expands the master key into the round keys using the `SubWord` and `RotWord` transformations.

Selection function: The 8-bit selection function we used in our experiments targets the result of the S-box lookup in the first round of encryption.

Table 4. Properties of the S-boxes of four analyzed ciphers. The values of TO, ITO, and SNR have a similar behavior as the value of NL for different S-boxes, but they have a different granularity. Thus, the study of NL with respect to CPA holds also for TO, ITO, and SNR, which are variations of NL.

Cipher	S-box	NL	TO	ITO	SNR
AES	S	112	7.860	6.916	9.600
LBlock	s_0	4	3.667	2.567	2.946
	s_1	4	3.667	2.567	2.807
	s_2	4	3.667	2.567	2.807
	s_3	4	3.667	2.567	2.946
	s_4	4	3.667	2.567	2.946
	s_5	4	3.667	2.567	2.807
	s_6	4	3.667	2.567	2.946
	s_7	4	3.667	2.567	2.946
Piccolo	S	4	3.667	2.567	3.108
PRINCE	S	4	3.400	2.333	2.129

Fantomas: Fantomas [17] is the non-involutive instance of a newly-crafted class of lightweight block ciphers, called LS-designs, that is specialized towards efficient Boolean masking. LS-designs facilitate the masking countermeasure to protect against DPA attacks by combining a linear diffusion layer in the form of a lookup table (L-box) with a bitsliced confusion layer. The 8-bit bitsliced S-box is an unbalanced Feistel network built from a 3-bit and a 5-bit S-box as in MISTY [28]. On the other hand, the 16-bit L-box has a branch number of 8 as explained in [17, Sect. 2.2] and was built from a systematic generator of the Reed-Muller code $RM(2,5)$. Fantomas does not have a key schedule. The family of LS-designs was very recently extended to XLS-designs [20], which aim to improve the security margins while retaining the implementation efficiency.

Selection function: Because there are four possible 8-bit inputs for the same MSB or LSB of the output of the mentioned 16-bit L-boxes, we had to attack both the MSB and LSB to recover the key. The selection function targets the inverse linear layer at the first round of decryption.

LBlock: LBlock [37] is based on a Feistel structure with a 64-bit block and an 80-bit key. At each round, the left branch goes through the round function F, while the right branch is rotated by 8 to the left. The two Feistel branches are swapped after each round, except for the last one. F consists of a substitution layer applied to the permutation of the left branch XORed with the round key. The confusion function includes eight 4-bit S-boxes used in parallel, while the diffusion function is defined as a permutation of eight 4-bit words. This 4-bit permutation can be implemented efficiently in both hardware and software environments. The key schedule of LBlock is designed in the form of a stream cipher and uses a left-rotation by 29 bits, two 4-bit S-boxes, and an XOR.

Selection function: The 4-bit selection function is given by the result of the substitution layer at the first round of encryption.

Piccolo: Piccolo [35] is a 64-bit block cipher supporting 80-bit and 128-bit keys and is suitable for restricted environments thanks to its high efficiency in hardware. It has a generalized Feistel structure with four 16-bit branches and a permutation-based key schedule. Due to its involution property, Piccolo can support decryption with little extra cost. The round function is very light and consists of two S-box layers, separated by a diffusion matrix. Piccolo also uses an 8-bit word-based permutation between rounds to improve diffusion.

Selection function: The 4-bit selection function targets the result of the first substitution layer of the first round function of encryption.

PRINCE: PRINCE [6] is a 64-bit block cipher with 128-bit keys based on the so-called FX construction. It is optimized for latency when implemented in hardware and allows the encryption of data within one clock cycle. PRINCE is suitable for pervasive applications with real-time security needs. The overhead for decryption on top of encryption is negligible due to the α-reflection property: decryption for one key corresponds to encryption with a related key. The key schedule expands the 128-bit key k to 192 bits, out of which the first two 64-bit subkeys $k_0, k_0^{'}$ are used as whitening keys, while the third subkey k_1 is used as a round key for the 12-round cipher called $PRINCE_{core}$. Each round of $PRINCE_{core}$ comprises an S-box layer, a linear layer, an addition of a round constant, and a key addition.

Selection function: The 4-bit selection function we used targets the substitution layer applied to the initial state XORed with the whitening key k_0 and round key k_1 at the first round of $PRINCE_{core}$. Thus, the attacker recovers the key $k^* = k_0 \oplus k_1$.

RC5: The RC5 [32] encryption algorithm is a Feistel-based cipher suitable for hardware and software implementation. A distinguishing feature of RC5 is the use of data-dependent rotations as a source of cryptographic strength. The rotation distance depends on the input data and is not predetermined. RC5 is parameterized and supports many implementation options; RC5-$w/r/b$ denotes a variant that operates on $2w$-bit blocks, has a key size of b bits, and performs r rounds. The encryption operation uses only three simple operations: addition modulo 2^w, bitwise XOR, and rotation to the left. However, the key expansion is quite complex and has a certain amount of "one-wayness." In our evaluation we used the same instance of RC5 as in [13], namely RC5-32/20/16, which is RC5 with 32-bit words, 20 rounds, and a 16-byte key.

Selection function: The selection function for RC5 targets the modular addition of the round key before the first encryption round. To avoid correlations with the reading the round key from memory instead of modular additions, we wrote the selection function in Assembly language to measure just the leakage generated by the targeted operation.

Simon: Simon [2, 3] is a family of lightweight block ciphers tuned for good performance in hardware, without sacrificing the performance in software too much. Simon $2n/mn$ denotes a Simon instance with a $2n$-bit block size and an mn-bit key, where n can be 16, 24, 32, 48, or 64. Designed to be very small in hardware and easy to serialize at many levels, it uses an extremely simple and low-complexity round function, which employs bitwise AND, bitwise XOR, and circular shifts applied to n-bit data words. The nonlinearity is provided by the bitwise AND. Simon's key schedule uses a sequence of 1-bit round constants to eliminate slide properties and circular shift symmetries.

Selection function: To increase leakage, we attacked the composition of the XOR and AND operations at the end of the first round of decryption because at that time the intermediate value is written to memory.

Speck: Speck [2, 3] is a family of software-optimized block ciphers that is also efficient in hardware. An instance with a $2n$-bit block and a mn-bit key is referred to as Speck $2n/mn$, where n can be 16, 24, 32, 48, or 64. The round function comprises bitwise XOR, addition modulo 2^n, and rotations applied to n-bit data. Speck gets its nonlinearity solely from the modular additions. The key schedule uses the encryption round function to generate the round keys.

Selection function: The used selection function gives the result of the modular subtraction of the two Feistel branches in the first decryption round. The attacker can take advantage of the memory-write operation of the result of the selection function rotated by 8 bits to the left.

6 Experimental Results

We distinguish between two main classes of lightweight ciphers with respect to their implementations' resistance against CPA. The first class contains ciphers that are implemented using lookup tables, while the second class comprises the ARX-based designs, whose operations generally leak less than table lookups.

First class: The first class can be further divided into three different categories of ciphers. The *first category* contains the AES, whose 8-bit S-box leaks much more than any other considered selection function. Our attacks required only 59 power traces to recover the four key-bytes with 100% success rate. The *second category* consists of the three lightweight ciphers LBlock, Piccolo, and PRINCE, each using one or more 4-bit S-boxes for the substitution layer. All members of this category leak enough information to make the recovery of the key with a small number of traces possible. On average, a little bit more than 100 traces were enough to get the subkeys of these ciphers with 100% success rate. However, two subkeys of LBlock and two subkeys of Piccolo required a lot more traces since the sensitive results of the selection functions are not written to memory after the targeted operation and hence the attacker correlates the reading of the S-box content (i.e. `ld` instruction) instead of the writing of the S-box output (i.e. `st` instruction). The *third category* is represented by ciphers that use linear lookup tables, e.g. Fantomas. Our attack against the implementation of Fantomas is a multi-target attack [26] because a normal attack failed to recover two bits of each attacked subkey. The multi-target attack enabled us to reveal the four key-bytes using 165 traces with 100% success rate.

Second class: The second class covers the ARX designs RC5, Simon, and Speck, for which we were not able to recover the full secret key due to reduced leakage. If we consider, for example, the attacks to obtain the fourth key byte

$k^* = \text{0x67}$ using $q = 2000$ traces, our experiments for RC5 and Simon gave a mean guessing entropy $\overline{GE}$ of 1.58 and 3.05, respectively. However, in the case of Speck, we managed to reveal k^* using 1345 traces with 100% success rate.

The Assembly code generated from the C implementations of these ciphers executes four consecutive `st` instructions, which entails signal superposing. We tried to "cancel" this effect by reducing the frequency of the processor, but we had no success. Although the insertion of `nop` instructions between the stores improved the results, we decided to not use these modified implementations in our experiments because they give the attacker an unreasonable advantage and affect therefore the fair comparison with the ciphers from the first class.

Given the small size of the state of the ARX designs and the rather simple operations they carry out, we investigated the possibility of keeping the whole state in registers during the entire encryption process. The 64-bit block version of both Simon and Speck can be implemented in Assembly without having to execute a single `st` instruction between the start and the end of the encryption operation. This approach significantly reduces the amount of leakage available to the attacker. But this leakage reduction optimization can not be applied to 128-bit block implementations of RC5, Simon, and Speck due to the restricted register space available on an 8-bit microcontroller. For RC5, we also tried the butterfly attack proposed in [38] on the modular addition, but the results were worse than when using the classical CPA attack.

We performed the described attacks also with a "low-cost" setup consisting of an Arduino Uno board and an Analog Discovery oscilloscope with a built-in differential probe. The Arduino board gets its supply voltage through an USB connection, which is also used for the communication with the computer that controls the trace acquisition process. We did not employ any noise reduction techniques. The experiments with the low-cost setup produced similar results for the ciphers in the first class, except for Fantomas, but required more traces due to the increased noise levels. For example, the AES key could be recovered with 80% success rate using 36 power traces with the first setup, but 58 traces were necessary with the second (i.e. low-cost) setup. Similarly, to retrieve the PRINCE key with the same success rate, the first setup needed 65 traces, while the second setup required 85 traces. For the ciphers from the second class, the low-cost setup yielded much worse results. When using 5000 traces, the mean guessing entropy for the attack against RC5 increased from 3.68 (low noise) to 22.29 (high noise). Similarly, for Simon we got $\overline{GE} = 9.97$ in the noise-reduced setting and $\overline{GE} = 16.44$ with the cheap equipment.

All our experiments were conducted on unprotected implementations of the ciphers. However, many security-critical applications require countermeasures against SCA attacks, e.g. masking. In this context, it is known that linear and Boolean operations, such as those performed by Fantomas, RC5, Simon, and Speck, can be masked with relatively low overheads in terms of execution time and code size. On the other hand, masking a nonlinear S-box like that of AES generally entails a significant performance and code-size penalty. Somewhere in the middle between these two extremes are LBlock, Piccolo, and PRINCE.

7 Conclusions

Following a practical approach, we investigated the leakage of various selection functions widely used in existing lightweight ciphers for an 8-bit processor. We analyzed how these results relate to the intuition about side-channel leakages based on the nonlinearity of the selection function. Thereby, we identified three imperfections of leakage evaluation based on nonlinearity, namely for AND and OR bitwise operations, for 4-bit S-boxes, and for linear lookup tables.

Using the knowledge gained from the evaluation of selection functions, we attacked unprotected software implementations of eight well-known lightweight ciphers, namely AES, Fantomas, LBlock, Piccolo, PRINCE, RC5, Simon, and Speck. We grouped the results of our experiments into two classes according to the observed resistance against CPA attacks. The unprotected implementation of AES was broken with the smallest number of power traces, followed by the implementations of lightweight ciphers using 4-bit S-boxes, and thereafter the implementation of Fantomas, whose L-boxes required slightly more traces than the 4-bit S-boxes. On the other hand, the ARX-based designs RC5, Simon, and Speck leaked less as we could not recover the full key for any of them. We also demonstrated that different implementation options can increase the resilience of lightweight block ciphers against power analysis attacks.

The software implementations of the three ARX designs we considered are characterized by a certain level of "intrinsic" resilience against CPA. They can also be efficiently masked with relatively small impact on execution time and code size. These features make ARX constructions excellent candidates for the implementation of lightweight block ciphers for the IoT.

Acknowledgements. We thank Yann Le Corre and André Stemper for their help with the measurement setup. The work of Daniel Dinu is supported by the CORE project ACRYPT (ID C12-15-4009992) funded by the Fonds National de la Recherche (FNR) Luxembourg.

References

1. V. Banciu, E. Oswald, and C. Whitnall. Exploring the Resilience of Some Lightweight Ciphers Against Profiled Single Trace Attacks. In *Constructive Side-Channel Analysis and Secure Design*, pages 51–63. Springer, 2015.
2. R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, and L. Wingers. The SIMON and SPECK Families of Lightweight Block Ciphers. *IACR Cryptology ePrint Archive*, 2013.
3. R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, and L. Wingers. SIMON and SPECK: Block Ciphers for the Internet of Things. In *NIST Lightweight Cryptography Workshop*, 2015.
4. O. Benoît and T. Peyrin. Side-Channel Analysis of Six SHA-3 Candidates. In *Cryptographic Hardware and Embedded Systems–CHES 2010*, pages 140–157. Springer, 2010.

5. S. Bhasin, T. Graba, J.-L. Danger, and Z. Najm. A Look into SIMON from a side-channel perspective. In *IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, pages 56–59. IEEE, 2014.
6. J. Borghoff, A. Canteaut, T. Güneysu, E. B. Kavun, M. Knezevic, L. R. Knudsen, G. Leander, V. Nikov, C. Paar, C. Rechberger, P. Rombouts, S. S. Thomsen, and T. Yalçin. PRINCE–A Low-latency Block Cipher for Pervasive Computing Applications. In *Advances in Cryptology–ASIACRYPT 2012*, pages 208–225. Springer, 2012.
7. E. Brier, C. Clavier, and F. Olivier. Correlation Power Analysis with a Leakage Model. In *Cryptographic Hardware and Embedded Systems–CHES 2004*, pages 16–29. Springer, 2004.
8. C. Carlet. On highly nonlinear S-boxes and their inability to thwart DPA attacks. In *Progress in Cryptology–INDOCRYPT 2005*, pages 49–62. Springer, 2005.
9. K. Chakraborty, S. Maitra, S. Sarkar, B. Mazumdar, D. Mukhopadhyay, and E. Prouff. Redefining the Transparency Order. Cryptology ePrint Archive, Report 2014/367, 2014.
10. S. Chari, J. R. Rao, and P. Rohatgi. Template Attacks. In *Cryptographic Hardware and Embedded Systems–CHES 2002*, pages 13–28. Springer, 2003.
11. CryptoLUX Team. FELICS – Fair Evaluation of Lightweight Cryptographic Systems. `https://www.cryptolux.org/index.php/FELICS`, 2015.
12. J. Daemen and V. Rijmen. *The Design of Rijndael: AES-The Advanced Encryption Standard.* Springer Science & Business Media, 2013.
13. D. Dinu, Y. Le Corre, D. Khovratovich, L. Perrin, J. Großschädl, and A. Biryukov. Triathlon of Lightweight Block Ciphers for the Internet of Things. Cryptology ePrint Archive, Report 2015/209, 2015. `http://eprint.iacr.org/`.
14. D. Evans. The Internet of Things: How the Next Evolution of the Internet is Changing Everything. Cisco IBSG white paper, available for download at `http://www.cisco.com/web/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf`, Apr. 2011.
15. B. Gérard, V. Grosso, M. Naya-Plasencia, and F.-X. Standaert. Block Ciphers that are Easier to Mask: How Far Can we Go? In *Cryptographic Hardware and Embedded Systems–CHES 2013*, pages 383–399. Springer, 2013.
16. B. Gierlichs, L. Batina, P. Tuyls, and B. Preneel. Mutual Information Analysis. In *Cryptographic Hardware and Embedded Systems–CHES 2008*, pages 426–442. Springer, 2008.
17. V. Grosso, G. Leurent, F.-X. Standaert, and K. Varıcı. LS-designs: Bitslice Encryption for Efficient Masked Software Implementations. In *Fast Software Encryption*, pages 18–37. Springer, 2014.
18. S. Guilley, P. Hoogvorst, and R. Pacalet. Differential Power Analysis Model and Some Results. In *CARDIS*, pages 127–142. Springer, 2004.
19. S. Guilley, P. Hoogvorst, R. Pacalet, and J. Schmidt. Improving side-channel attacks by exploiting substitution boxes properties. In *International Workshop on Boolean Functions: Cryptography and Applications*, pages 1–25, 2007.
20. A. Journault, F.-X. Standaert, and K. Varici. Improving the Security and Efficiency of Block Ciphers based on LS-Designs. *Designs, Codes and Cryptography*, 2016.
21. M. Joye and F. Olivier. Side-channel analysis. *Encyclopedia of Cryptography and Security*, pages 1198–1204, 2011.
22. T. Kasper, D. Oswald, and C. Paar. Sweet dreams and nightmares: Security in the Internet of things. In *Information Security Theory and Practice: Securing the Internet of Things*, pages 1–9. Springer, 2014.
23. P. Kocher, J. Jaffe, and B. Jun. Differential Power Analysis. In *Advances in Cryptology–CRYPTO'99*, pages 388–397. Springer, 1999.

24. P. C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In *Advances in Cryptology–CRYPTO'96*, pages 104–113. Springer, 1996.
25. K. Lemke, K. Schramm, and C. Paar. DPA on n-bit Sized Boolean and Arithmetic Operations and Its Application to IDEA, RC6, and the HMAC-construction. In *Cryptographic Hardware and Embedded Systems–CHES 2004*, pages 205–219. Springer, 2004.
26. L. Mather, E. Oswald, and C. Whitnall. Multi-target DPA Attacks: Pushing DPA Beyond the Limits of a Desktop Computer. In *Advances in Cryptology–ASIACRYPT 2014*, pages 243–261. Springer, 2014.
27. M. Matsui. Linear Cryptanalysis Method for DES Cipher. In *Advances in Cryptology–EUROCRYPT'93*, pages 386–397. Springer, 1994.
28. M. Matsui. New Block Encryption Algorithm MISTY. In *Fast Software Encryption*, pages 54–68. Springer, 1997.
29. NIST. Advanced Encryption Standard (AES). Federal Information Processing Standards Publication (FIPS) 197, 2001.
30. G. Piret, T. Roche, and C. Carlet. PICARO–A Block Cipher Allowing Efficient Higher-Order Side-Channel Resistance. In *Applied Cryptography and Network Security*, pages 311–328. Springer, 2012.
31. E. Prouff. DPA Attacks and S-boxes. In *Fast Software Encryption*, pages 424–441. Springer, 2005.
32. R. L. Rivest. The RC5 Encryption Algorithm. In *Fast Software Encryption*, pages 86–96. Springer, 1995.
33. R. Selvam, D. Shanmugam, and S. Annadurai. Vulnerability Analysis of PRINCE and RECTANGLE using CPA. In *ACM Workshop on Cyber-Physical System Security*, pages 81–87, 2015.
34. D. Shanmugam, R. Selvam, and S. Annadurai. Differential Power Analysis Attack on SIMON and LED Block Ciphers. In *Security, Privacy, and Applied Cryptography Engineering*, pages 110–125. Springer, 2014.
35. K. Shibutani, T. Isobe, H. Hiwatari, A. Mitsuda, T. Akishita, and T. Shirai. Piccolo: An Ultra-Lightweight Blockcipher. In *Cryptographic Hardware and Embedded Systems–CHES 2011*, pages 342–357. Springer, 2011.
36. F.-X. Standaert, T. G. Malkin, and M. Yung. A Unified Framework for the Analysis of Side-Channel Key Recovery Attacks. In *Advances in Cryptology–EUROCRYPT 2009*, pages 443–461. Springer, 2009.
37. W. Wu and L. Zhang. LBlock: A Lightweight Block Cipher. In *Applied Cryptography and Network Security*, pages 327–344. Springer, 2011.
38. M. Zohner, M. Kasper, and M. Stöttinger. Butterfly-Attack on Skein's Modular Addition. In *Constructive Side-Channel Analysis and Secure Design*, pages 215–230. Springer, 2012.
39. M. Zohner, M. Kasper, M. Stöttinger, and S. Huss. Side channel analysis of the SHA-3 finalists. In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1012–1017. IEEE, 2012.

A Additional Tables

Table 5. Detailed leakages for different selection functions φ_i as defined in Table 2.

Selection function	Correct key								
	0x00	0x01	0x03	0x07	0x0F	0x1F	0x3F	0x7F	0xFF
φ_1	-0.225	0.098	0.086	0.057	-0.031	-0.052	-0.001	0.011	0.007
φ_2	0.006	-0.005	-0.002	-0.073	-0.002	0.026	0.015	0.072	-0.202
φ_3	-0.145	-0.160	-0.173	-0.190	-0.167	-0.152	-0.142	-0.125	-0.124
φ_4	0.129	0.134	0.134	0.127	0.150	0.125	0.117	0.096	0.131
φ_5	0.121	0.120	0.147	0.125	0.113	0.109	0.111	0.141	0.110
φ_6	0.597	0.582	0.578	0.577	0.566	0.595	0.603	0.586	0.593
φ_7	0.341	0.343	0.338	0.354	0.337	–	–	–	–
φ_8	0.234	0.223	0.228	0.249	0.230	0.245	0.244	0.233	0.234
φ_9	0.319	0.331	0.361	0.350	0.338	–	–	–	–
φ_{10}	0.252	0.245	0.264	0.256	0.263	0.268	0.264	0.255	0.268
φ_{11}	0.265	0.257	0.273	0.273	0.278	–	–	–	–
φ_{12}	0.139	0.135	0.146	0.143	0.136	0.142	0.129	0.145	0.131
φ_{13}	0.094	0.089	0.079	0.061	0.061	0.080	0.105	0.099	0.120
φ_{14}	0.036	0.027	0.026	0.028	0.018	0.047	0.060	0.062	0.069
φ_{15}	0.144	0.121	0.137	0.127	0.129	0.145	0.134	0.151	0.143
φ_{16}	0.078	0.073	0.072	0.037	0.074	0.093	0.120	0.100	0.100

Table 6. Nonlinearity (NL) of the components of the modular addition (selection functions φ_4 and φ_5 from Table 2). By nonlinearity of a component of an (n, m) function F, we mean the nonlinearity of F computed for a fixed vector $v \in \mathbb{F}_2^{m*}$ as in Equation (1). "Number" denotes how many components have the given nonlinearity NL, "Proportion (%)" is the proportion of the given nonlinearity NL with respect to the nonlinearity of all components of F.

(a) $\varphi_4 : \mathbb{F}_2^{16} \mapsto \mathbb{F}_2^8$, $\varphi_4(x, k) = x \boxplus k$

NL	Number	Proportion (%)
0	1	0.39
16384	26	10.20
24576	100	39.22
28672	112	43.92
30720	16	6.27

(b) $\varphi_5 : \mathbb{F}_2^{17} \mapsto \mathbb{F}_2^8$, $\varphi_5(x, k, c) = x \boxplus k \boxplus c$

NL	Number	Proportion (%)
0	1	0.39
32768	26	10.20
49152	100	39.22
57344	112	43.92
61440	16	6.27