A Tale of Snakes and Horses: Amplifying Correlation Power Analysis on Quadratic Maps

Anna Guinet[1] , Georg Land[1] , Ioan Gabriel Bucur[2] and Tim Güneysu[1]

[1] Ruhr-Universität Bochum, Bochum, Germany
email@annagui.net,mail@georg.land,tim.gueneysu@rub.de

[2] Radboud University Nijmegen, Nijmegen, The Netherlands

Abstract. We study the success probabilities of two variants of Correlation Power Analysis (CPA) to retrieve multiple secret bits. The target is a permutation-based symmetric cryptographic construction with a quadratic map as an S-box. More precisely, we focus on the nonlinear mapping χ used in the Xoodoo and Keccak-p permutations, which is affine equivalent to the nonlinear mapping of Ascon. We thus consider three-bit and five-bit S-boxes. Our leakage model is the difference in power consumption of register cells before and after one round. It reflects that, in hardware, the aforesaid cryptographic algorithms are usually implemented by deploying a round-based architecture. The power consumption difference depends on whether the targeted bits in the register flip. In particular, we describe two attacks based on the CPA methodology. First, we start with a standard CPA approach, for which, to the best of our knowledge, we are the first to point out the differences between attacking a three-bit and a five-bit S-box. For CPA, the highest correlation coefficient is the most likely secret hypothesis. We improve on this with our novel combined Correlation Power Analysis (combined CPA), or *Snake* attack, which uses quadratic map cryptanalysis (e.g., of the function χ) to achieve a better attack in terms of the number of traces required and computational complexity. For the Snake attack, sums of absolute or squared values of correlation coefficients are used to determine the most likely guess. As a result, we effectively show that our proposed Snake attack can recover the secret in $n2^2$ (or $n2^2 + n$) intermediate results, compared to 2^{2n} for the CPA, where n is the length of the targeted S-Box. We collected power measurements from a hardware setup to demonstrate practical attack success probabilities according to the rank of the correct secret hypothesis both for Xoodoo and Keccak-p. In addition, we explain our success probabilities thanks to the Henery model developed for horse races. In short, after performing 16,896 attacks, the Snake attack or combined CPA on Xoodoo consistently recovers the correct secret bits with each attack using 43,860 traces on average and with only 12 correlations, compared to 61,380 traces for the standard CPA attack with 64 correlations. The Snake attack requires about one-fifth as many correlation values as the standard CPA. For Keccak-p, the difference is more drastic: the Snake attack recovers the secret bits invariably after 771,600 traces with just 20 correlations. In contrast, the standard CPA attack operates on 1,024 correlations (about fifty times more than the Snake attack) and requires 1,223,400 traces.

Keywords: correlation power analysis · hardware testbed · success probability · ranking · quadratic S-boxes

1   Introduction

Side-channel analysis enables the exploitation of weaknesses in the physical implementation of cryptographic primitives that may leak secret data. Indeed, the physical behavior of an electronic device, e.g., its power consumption or electromagnetic radiation, carries information about the processed data and internal operations. An adversary that can measure such a physical leakage presents a significant threat to devices that manipulate sensitive information. More precisely, by analyzing the power consumption, they[1] can retrieve parts of the secret with methods like Differential Power Analysis (DPA) [KJJ99] or Correlation Power Analysis (CPA) [BCO04]. If countermeasures are insufficient, an adversary could *practically* decrypt messages or forge an authentication code, although the primitive is secure. The cryptographic community, therefore, aims to find sound countermeasures [BDPVA10, BNN+15, ABP+18, PAM19, ZSS+21].

In this work, we investigate, with CPA, the security of a device that implements a permutation-based cryptographic algorithm with an S-box based on a quadratic map, i.e., has algebraic degree two. An S-box is the nonlinear layer of a round function in a symmetric cryptographic algorithm. We chose to study quadratic maps partly because research work aimed at classifying n-bit S-boxes in affine equivalence classes to find optimal S-boxes found an interest in quadratic maps ($n \in \mathbb{N}$, $n \in [1, 6]$) [BBS17, DC07, LP07, ZBRL15, MB19]. Moreover, they are reasonably simple to manipulate for our proposed CPA attacks, which exploit the algebraic expression of the S-box.

Our electronic device has a register that stores intermediate data that is updated after each of the aforementioned rounds. That is because permutation-based algorithms are implemented with a round-based architecture. We aim to quantify the gain in attack success probability between exploiting the leakage of multiple storage cells of a register state at once and the combination of the individual leakage of those cells for different S-box sizes. The first attack corresponds to the standard CPA, and the second is our newly proposed combined CPA strategy. The aim is to explore the difference in success probability and computational complexity between those two attacks, for small S-boxes, with simple assumptions.

We focus on the function χ, as it was shown in [Dae95] to be the quadratic function with the most interesting properties to provide resistance against differential and linear cryptanalysis. The function χ is used in the Keccak-*p* [BDPV11] and Xoodoo [DHVV18] cryptographic permutations, which are at the core of cryptographic algorithms such as Keccak/SHA-3 [Dwo15], Keyak [BDP+16b], Ketje [BDP+16a], Xoofff [DHVV18] or Xoodyak [DHP+20] NIST Lightweight Cryptography competition finalist. The function operates on a group of five bits in Keccak-*p* and on one of three bits in Xoodoo. We apply thus our attacks on five-bit and three-bit S-boxes, which are instances of the mapping χ.

Contribution. Earlier work by Bertoni *et al.* [BDD+12] considered the security of the function χ for a single register cell against DPA using simulations. They studied the power consumed to retrieve two bits of the secret after one round of the Keccak-*p* permutation. Samwel and Daemen [SD17] extended this work by targeting the five secret bits of a Keyak (Lake Keyak) S-box with DPA.

- We extend and formalize the previous analyses by proposing a methodology to target n register bits, which are iteratively updated by a cryptographic round function. For that purpose, we improve the leakage model from [BDD+12, SD17] in Section 3 to adapt it to the Gaussian additive noise model [MOS11] for our round-based hardware architecture (Section 2). Additionally, Samwel and Daemen [SD17] proposed to partition the power consumption values according to the values of the messages in

---

[1]We use the singular 'they' as a gender-neutral pronoun for the attacker throughout the paper in an effort to be inclusive.

order to exploit the leakages of several register bits: we formalize and generalize this idea to any S-box size that uses a quadratic map.

- In Section 4, the standard CPA attack is detailed for the first time for the quadratic map χ, to the very best of our knowledge. We further present the combined CPA strategy, which aggregates individual leakage information from multiple register bits by exploiting the cryptanalysis of the function χ. We nickname it the 'Snake attack' because of the visual representation of the combination of individual leakage bit results to retrieve several secret bits.

- In Section 5, we present a method for determining the rank of the solution secret hypothesis for an attack strategy, and derive from it the probability of success.

- We provide practical success probabilities in Section 6 for our hardware setup. We argue that our results follow the probability Henery model [Hen81], which was originally proposed to model horse races.

In conclusion, for our three-bit and five-bit S-box, the combined CPA, or Snake, attack has a higher probability of success than CPA for the same number of power traces, i.e., power consumption measurements, while, most importantly, having fewer correlation results to manipulate. Indeed, after performing 16,896 attacks, combined CPA on a three-bit S-box unfailingly recovers the secret solution bits starting from 43,860 power traces with simply 12 correlation values, compared with 64 correlation results and 61,380 power traces for CPA. The difference is even more noticeable for a five-bit S-box, since combined CPA consistently determines the secret bits from 20 correlation results after 771,600 power traces. In contrast, the standard CPA attack only does it after 1,223,400 power traces and requires 1,024 correlation results. In a nutshell, the Snake attack or combined CPA has the advantage of manipulating fewer intermediate steps at once than the classic CPA to ease the effort of side-channel analysts.

2 Hardware Architecture

We consider a hardware implementation of a permutation using a quadratic map as an S-Box for our CPA attacks. In the remainder of this paper, we focus on the well-known function χ, which is investigated for its interesting differential and correlation properties [Dae95, MPLJ19, DMM21], or more generally for its state diagram [SD23], to find potential distinguishers for cryptanalysts. Multiple cryptographic schemes are based on permutations like Keccak-p [BDPV11, Dwo15] or Xoodoo [DHVV18], which use the function χ to have a simple nonlinear layer. The previous permutations are well-studied in the cryptographic community [LSL21, LIMY20, LQT19, GPT21, EME22].

2.1 Permutation

We study the function χ in a permutation-based cryptographic scheme, namely the sponge construction [BDPV08]. This construction is an iterative construction that applies a permutation f several times. The permutation consists in itself of a round function that is applied iteratively to an array of bits or *state*. The round function contains a linear layer λ and a nonlinear one. The latter comprises mappings χ_n in parallel on arrays of n bits.

Definition 1 (Function χ_n). Let n be an odd integer. Let $i \in \mathbb{Z}/n\mathbb{Z}$ be an index to designate the i-th bit in a binary sequence of length n. The function $\chi_n \colon \mathbb{F}_2^n \to \mathbb{F}_2^n$ restricted to a n-bit array and applied on the bit x_i is

$$\chi_n(x) \upharpoonright_i = x_i \oplus (x_{i+1} \oplus 1)\, x_{i+2}\,.$$

The length n of a sequence to which χ_n is applied, depends on the permutation used in the cryptographic scheme. We use the generic term *sequence* to refer to the said n bits to be permutation-agnostic.

Xoodoo. The permutation Xoodoo [DHVV18] applies a round function to a 384-bit state consisting of of 128 columns of three bits. The round function of Xoodoo is decomposed in two linear mappings θ and ρ_{west}, followed by the addition of a round constant ι, then the application of the nonlinear mapping χ, and another linear mapping ρ_{east}. The nonlinear mapping χ is applied to the individual bits x_i of a *column*. Therefore, in our notation, it corresponds to applying the function χ_3 on a binary sequence of three bits.

Keccak-*p*. The permutation Keccak-*p*, as specified by the NIST SHA-3 standard [Dwo15], is defined over a state of $5 \cdot 5 \cdot 2^l$ bits with $l \in \mathbb{N}$, $l \in [0, 6]$, with a number of rounds being equal to $12 + 2l$. The state is organized in an array of five rows times five columns times 2^l lanes. The round function of Keccak-*p* is composed of five mappings: three linear mappings θ, ρ, and π are first applied, followed by the nonlinear mapping χ and the round constant addition ι. The nonlinear mapping χ is applied on the bits x_i of a *row*, which corresponds to the application of χ_5 on a binary sequence of five bits in our notation.

Henceforth, we consider five-bit and three-bit sequences to focus on χ_3 and χ_5. As an additional remark, we focus on the function χ in this paper, not only for its simplicity and its interesting properties for symmetric cryptography but also because we know that all invertible quadratic functions operating on three bits are affine equivalent[2] to χ_3 if all their three functions restricted to one bit are quadratic. Therefore, this nonlinear function is the best candidate to test and present our CPA attacks on small S-boxes. On a side note, according to De Cannière [DC07], there are three affine equivalence classes for this size of S-box. They are made explicit in the work of Bilgin *et al.* [BNN+15]. A well-known example used in symmetric cryptography for a slightly larger S-box is the five-bit S-box of Ascon [DEMS21], winner of the NIST Lightweight Cryptography competition, which is affine equivalent to the S-box of Keccak.

2.2   Round-Based Hardware Architecture

We consider an electronic device with a round-based cryptographic algorithm implemented without protection. Its hardware architecture comprises a register composed of storage cells. The round logic performs the round function on the register state and updates it with the output of the round function, as depicted in Figure 1. This circuit computes an r-round permutation in r cycles. We keep the architecture as simple as possible to better understand how to optimize standard power attacks in such an elementary setup.
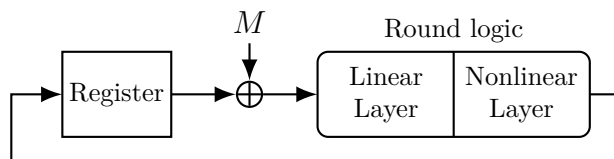
Figure 1: The round-based hardware architecture.

Before performing the first round, it is assumed that the register contains a secret K that results from earlier processing, e.g., the absorption of a key. Then, a message M is added to the register state once before applying the first round function. The message M has the same length as the secret K. The sum of the secret and the message $M \oplus K$ is

---

[2]Two $n \times n$-bit invertible S-boxes S_1 and S_2 are affine equivalent if there exists two affine invertible mappings A and B satisfying $A \circ S_1 \circ B = S_2$. The affine mappings can be written as $A \cdot x + a$ where A is a $n \times n$ invertible matrix over $\mathbb{F}_2$ and a is a n-bit constant [DC07].

the input of the round logic. The linear layer λ, which we do not detail for the sake of generality, is applied to this value to generate the input to the nonlinear layer. We then define two intermediate variables μ and κ in order to set-up our attack.

Definition 2 (μ and κ)**.** Let μ and κ be, respectively, the n bits of $\lambda(M)$ and $\lambda(K)$ that are input to the targeted sequence χ_n.

The input of the considered sequence χ_n is $\mu \oplus \kappa$ with κ constant to be more specific. The register stores the output of the round logic that equals $\chi_n(\mu \oplus \kappa)$ for one sequence χ_n after the first round. The presented architecture is an implementation without round constants for the sake of simplicity.

3   Correlation Power Analysis

Cryptographic operations are performed in electronic devices where data is stored in registers that are a collection of flip-flops. A flip-flop can be thought of as a cell that stores a binary digit. When the state of the flip-flop is changed, i.e., the stored bit is switched, a certain amount of power is applied to perform the change. In contrast, switching power is not required if the state is maintained.

The power consumption of an electronic device is a function of the binary data processed by the circuit, but in a complex manner. It has noisy behavior, partly due to the difficulty of accurately modeling the power consumption and inherent non-deterministic effects. However, the correlation between the power consumption and its switching activity can be measured and exploited to reveal some cryptographic secrets using a statistical attack, such as DPA [KJJ99] or CPA [BCO04]. For those attacks to work, the measured power consumption must depend on known data and constant secret bits.

We consider as an example the circuit presented in Section 2.2 and an attacker that targets one flip-flop that stores a single bit of the state. This bit is updated after every round. The adversary performs a CPA attack to exploit the leakage due to the switching of the said flip-flop after one round. The stored bit is denoted by b_r, with r the round index. For the cipher, we can express the value of b_r after one round as a simple algebraic expression that depends on a known message and a constant secret at the input. Then, the adversary measures the power consumption for many cryptographic computations for the same secret but with different, known messages. A sequence of instantaneous power consumption measurements for a single message is commonly called a *power trace*. The adversary then computes $b_1 \oplus b_0$ for each power trace with the known input message and with a hypothesis on the secret bits. Each of these hypotheses now has an array of bits representing whether the target bit changed or not. If the secret hypothesis is correct, the value $b_1 \oplus b_0$ is also correct, and the traces where the bit flips (i.e., the sum results in 1) consume on average more power than the ones for which the bit does not flip (i.e., the sum is 0). Otherwise, for the incorrect hypotheses, the value $b_1 \oplus b_0$ usually does not correlate well with the power consumption, depending on how much nonlinearity is involved in the round function. In other words, the most likely secret hypothesis is the one that maximizes the correlation between the array of hypothetical bit (non-)flips and the power consumption.

The presented attack can also be generalized to target multiple bits at once. Intuitively, the more bits flip at once, the more power an electronic device consumes. Let b_r^i be the i-th bit in the r-th round. Then, for targeting ℓ bits at once, the adversary can use $\sum_{j=1}^{\ell} b_1^j \oplus b_0^j$ (for each hypothesis of the secret) to correlate with the power consumption. Again, the hypothesis that maximizes the correlation is deemed the correct one.

We aim to understand the CPA attack strategy that uses fewer power traces and computational complexity while targeting a quadratic map operating on a few bits to retrieve multiple secret bits and making simple hypotheses. More precisely, what is the

gain in success probability between extracting information from the power consumption of multiple register cells and combining those said cells individually for quadratic maps operating on different sequence sizes? To this end, we describe a leakage model based on [BDD+12].

3.1 Activity Function and Leakage Model

We assume the adversary knows the messages M_j that correspond to j specific traces and can compute μ for each message, which is the bits of $\lambda(M)$ that are input to a χ_n sequence (Definition 2). The attacker exploits the power consumption of m storage cells, with $m \leq n$, between the initial register state at time t_0 and the state after the first round at time t_1 where n is the length in bits of an S-box, i.e., binary input sequence. In this study, m is a parameter to determine whether we target the signal power consumption of individual bits ($m = 1$), or all the bits of the first sequence χ_n at a time ($m = n$ with $n \in \{3, 5\}$ in this paper).

Definition 3 (K')**.** Let K' be the m targeted bits of the register at time t_0 that stores the secret K. We denote by K_i' the i-th bit of K'.

Ultimately, the adversary aims to recover K', but depending on the attack strategy, they might only be able to recover κ at first. We now consider that the adversary aims to recover the secret (K', κ). The hardware architecture and attack parameters are depicted in Figure 2, where the targeted secret bits correspond to the first S-box in this example. Note that this is a widespread architecture for implementations of Xoodoo, and Keccak-*p* in Field-Programmable Gate Arrays (FPGAs) and Application-Specific Integrated Circuits (ASICs).
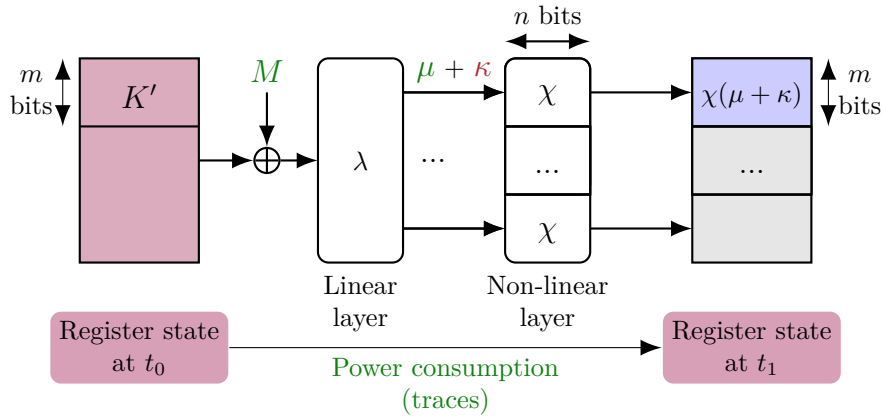

Figure 2: First round of the first permutation. The unknown variables and states to the attacker are in red, what the adversary knows is in green, and the scope of this study is in blue. The gray area contributes to the noise in our model.

We use a simple leakage model for the m register cells [BDD+12], sometimes denoted as the Hamming distance power consumption model [PSDQ05, BCO04]. It counts the number of bit flips between time t_0 and t_1. The power consumed to flip a bit value, i.e., to change a 0 into a 1 and vice versa, is assumed to be higher than keeping the same value over time. It also supposes that all cells contribute equally to the power consumption and that the transitions $0 \rightarrow 1$ and $1 \rightarrow 0$ lead to the same value. Under these assumptions, we define the activity function $d_i(K_i', \kappa, \mu) \in \mathbb{F}_2$ for one bit in Definition 4. If the bit value flips, it equals 1, otherwise, it equals 0. To compute the Hamming distance between the first m bits of the state at time t_0 and t_1, we sum the outputs of the activity function of

those bits in Definition 5. We thus obtain the power consumption of the first m bits; it is our *signal*. It should be noted that the round constants, which are not described here for the sake of simplicity, shall be included in the activity function in practice.

Definition 4 (Activity function). For $i \in \mathbb{Z}/n\mathbb{Z}$, let K_i' be the i-th bit of the secret register state. Let $d_i \in \mathbb{F}_2$ be the expression of the activity of the i-th bit after the first round:

$$d_i(K_i', \kappa, \mu) = K_i' \oplus \kappa_i \oplus \mu_i \oplus (\kappa_{i+1} \oplus \mu_{i+1} \oplus 1)(\kappa_{i+2} \oplus \mu_{i+2}).$$

Definition 5 (Signal power consumption). The signal power consumption $S \in \mathbb{R}$ depends on the m bits at time t_0 and the intermediate variables μ and κ,

$$m \in \{1, n\}: \quad S(K', \kappa, \mu) = \sum_{i=0}^{m-1} (-1)^{d_i(K_i', \kappa, \mu)}.$$

In Definition 5, if a bit x_i flips, the corresponding element of the sum is -1, and otherwise $+1$. By doing so, the model is balanced. Summing the elements $(-1)^{d_i(K_i', \kappa, \mu)}$ m times encodes the number of bit flips of the targeted m bits stored in the register stage. This definition of signal power consumption makes it possible to recover our secret in our measurements from the hardware setup, as it heavily depends on the secret values. If the adversary wants to target one bit of the secret, they choose $m = 1$ to capture the signal value of one bit; otherwise they take $m = n$ to recover all the secret bits K', which are inputs to the targeted S-box.

As we focus on one sequence χ_n at a time, we consider that the power consumption of the remaining sequences of χ_n contributes to the overall noise in our model. We define R_j as the noise power consumption of a trace j from the non-targeted sequences χ_n and other parts of the electronic device, in Definition 6.

Definition 6 (Noise power consumption). We assume the noise power consumption is a random variable drawn from a zero-mean normal distribution with variance σ^2 for a trace j. It is independent of the intermediate variables μ and κ:

$$R_j \sim \mathcal{N}(0, \sigma^2).$$

The power consumption P_j of a single trace is described with the Gaussian additive noise model [MOS11] in Definition 7. It is characterized by a random variable drawn from a normal distribution with variance σ^2 and mean $S \neq 0$.

Definition 7 (Power consumption). We define the power consumption for a secret (K', κ) to be the sum of the signal part and the noise part for a trace j and a message M_j, such as

$$P_j(K', \kappa, \mu) = S(K', \kappa, \mu) + R_j \quad \text{with} \quad \mu = \lambda(M_j).$$

The above definition makes it possible to model real experiments in a simple manner accurately. At last, we take advantage of the simplicity of the presented leakage model to propose a novel attack strategy to recover some secret bits by manipulating the activity function underlying the modeled power consumption. By doing so, we take advantage of the cryptanalysis of the S-box.

3.2 Pearson Correlation Coefficient and Signal Reference Values

In a standard CPA attack, the adversary first feeds an oracle (i.e., our activity function) with some knowledge (μ) and hypotheses about the secret. The oracle then estimates of the power consumption [BCO04], as explained earlier, and creates a set of power traces P containing power traces P_j. Following this, the adversary computes the Pearson correlation

coefficient between the a set of power traces P and the oracle output for each hypothesis. The more power traces the adversary has, the higher their chances are to recover the secret solution, thanks to the Gaussian additive noise model and Proposition 1. More precisely, the average power trace of a set P has a mean S and a variance that decreases proportionally with the number of measurements. As a result, the amount of power traces collected by the adversary is positively correlated with the probability of determining whether a bit has flipped for fixed (K', κ, μ).

Proposition 1 (Average of Gaussian random variables, see [LL02, Section 2.4]). *The average of N independent Gaussian random variables with mean S and variance σ^2 is also a Gaussian random variable with mean S and variance $\sigma^2 \cdot N^{-1}$.*

The knowledge of the oracle, or activity function, is stored in tuples of *signal reference values* S_{ref}, which are determined solely by the activity of (K', κ, μ). More specifically, the signal reference values result from the computation of the signal power consumption (Definition 5) for all possible values of (K', κ, μ) given the m targeted register bits and the length n of the S-box. They are first sorted in tuples depending on the value K'. Then, each tuple contains sub-tuples indexed by the corresponding value of κ, which themselves list signal reference values indexed by the corresponding value of the message μ. Suppose the adversary targets all the m bits of the register ($m = n$). In that case, there are $3 \cdot 2^n$ evaluations of the signal power consumption from Definition 5 for all the possible values of K', κ, and μ. Otherwise, if the adversary targets one bit of the register ($m = 1$), then the signal power consumption is evaluated for the two possible values of K' and 2^3 values of κ and μ due to the algebraic expression of the activity function (Definition 4), which depends on one bit of K', and three bits of κ and μ.

Definition 8 (Signal reference values S_{ref}). The signal power consumption values are stored in tuples of tuples $S_{\text{ref}}(K')$, one for each of the 2^m possibilities of K', which contains tuples $S_{\text{ref}_{K'}}(\kappa)$, one for each value of κ, which consists of all signal power consumption evaluations for the message μ with fixed (K', κ). More formally, we have:

$$S_{\text{ref}} \overset{\text{def}}{=} (S_{\text{ref}}(K') \mid K' \in \mathbb{F}_2^m),$$

which, for a fixed value K', consists of:

$$S_{\text{ref}}(K') = (S_{\text{ref}_{K'}}(\kappa) \mid \kappa \in \mathbb{F}_2^{m_{K'}}), \quad \text{with} \quad \begin{cases} m_{K'} = n & \text{if } m = n \, ; \\ m_{K'} = 3 & \text{if } m = 1. \end{cases}$$

Then, for a fixed secret value (K', κ), we have:

$$S_{\text{ref}_{K'}}(\kappa) = (S(K', \kappa, \mu) \mid \mu \in \mathbb{F}_2^{m_{K'}}),$$

where $S(K', \kappa, \mu)$ is the signal power consumption.

The adversary then computes the Pearson correlation coefficient between those reference values $S_{\text{ref}_{K'}}(\kappa)$ and the power consumption to find the most likely secret value of K' and κ, as presented in the following Definition.

Definition 9 (Pearson correlation coefficient). The Pearson correlation coefficient ρ of two random variables P and $S_{\text{ref}_{K'}}(\kappa)$ is

$$\rho_{P, S_{\text{ref}_{K'}}(\kappa)} = \frac{\mathbb{E}[P S_{\text{ref}_{K'}}(\kappa)] - \mathbb{E}[P]\,\mathbb{E}[S_{\text{ref}_{K'}}(\kappa)]}{\sigma_P\,\sigma_{S_{\text{ref}_{K'}}(\kappa)}},$$

with $\mathbb{E}$ the expectation and σ the standard deviation [LL02, Section 2.3].

Pearson correlation coefficients range between -1 and $+1$, where $+1$ (resp. -1) indicates that the two random variables are perfectly positively (resp. negatively) linearly correlated. In our case, it means that the power consumption is positively or negatively correlated with a secret value (K', κ). The adversary deduces which secret value is the most likely solution for the measured power traces by selecting the highest absolute correlation value.

The adversary's chance of finding a solution secret depends on the amount of gathered power traces. We further explore this fact to compute probabilities of success based on the number of collected traces for each of our attacks.

4 Attack Strategies

We detail two CPA attack strategies to target one sequence χ_n at a time. The first strategy is CPA [BCO04], which simultaneously targets n bits of the register. To the best of our knowledge, we are the first to provide the details of this attack on a quadratic map (χ_3 and χ_5 in this paper). In comparison, the second strategy combines n single-bit CPA attacks. We call it *combined CPA* or *Snake attack*. Both the presented strategies are device-agnostic and are finely tuned for quadratic maps.

4.1 Correlation Power Analysis

In our first strategy, the attacker exploits the joint power consumption of the n bits of the sequence χ_n at a time. They can thus have correlation results for 2^{2n} secret hypotheses at once. For example, for $n = 3$, the attacker can directly recover the six secret bits $(K'_0, K'_1, K'_2, \kappa_0, \kappa_1, \kappa_2)$. In order to exploit the leakage of the n register cells, the adversary employs the well-known CPA (see Section 3.2). We describe how to adapt it to our leakage model to compare it with combined CPA.

The adversary represents the signal power consumption values with 2^n tuples $S_{\text{ref}}(K')$, each containing 2^n tuples $S_{\text{ref}_{K'}}(\kappa)$, consisting of 2^n values for the possible values of μ (Definition 8). That is because the adversary computes signal power consumption reference values for $m = n$ (Definition 5). To be more comprehensive, we write below the signal reference tuples $S_{\text{ref}}(K')$ for three bits. The explicit values of the binary sequences are written in monospace font, e.g., 000. We denote by $S_{\text{ref}}(K' = \text{x})$ the tuple of signal reference values for a fixed K', whose value equals the sequence x. In a similar fashion, we write the sub-tuple of signal reference values for a fixed K' equal to x and κ to y by $S_{\text{ref}_\text{x}}(\kappa = \text{y})$:

$$S_{\text{ref}}(K' = \texttt{000}) = (S_{\text{ref}_{000}}(\kappa = \texttt{000}), S_{\text{ref}_{000}}(\kappa = \texttt{001}), \cdots, S_{\text{ref}_{000}}(\kappa = \texttt{111})),$$
$$S_{\text{ref}}(K' = \texttt{001}) = (S_{\text{ref}_{001}}(\kappa = \texttt{000}), S_{\text{ref}_{001}}(\kappa = \texttt{001}), \cdots, S_{\text{ref}_{001}}(\kappa = \texttt{111})),$$
$$\cdots$$
$$S_{\text{ref}}(K' = \texttt{111}) = (S_{\text{ref}_{111}}(\kappa = \texttt{000}), S_{\text{ref}_{111}}(\kappa = \texttt{001}), \cdots, S_{\text{ref}_{111}}(\kappa = \texttt{111})).$$

The tuples of signal references values for χ_5 can be trivially extended from the above tuples. An explicit example of a tuple of signal power references for a secret K' being 100 and $n = 3$ is provided in Table 1. In the table, each bit of μ is written as $\mu_0\mu_1\mu_2$, and the same goes for κ. The tuple $S_{\text{ref}}(K' = \texttt{100})$ is represented in a matrix form, where the columns are the said tuples $S_{\text{ref}_{100}}(\kappa)$ and the lines indicates the corresponding values of μ.

According to Definition 5, if all the n targeted bits of the register stage flip between time t_0 and t_1, the corresponding signal power consumption equals $-n$. If only one bit flips, it equals $+1$ for $n = 3$; if not, it equals -1. The signal power consumption is $+n$ only if the n bits have not flipped. Hence, the sign of the Pearson correlation coefficient is regarded to determine whether the bits have switched. The adversary thus has enough information to recover the $2n$ secret bits at once, given sufficient power traces. Ultimately,

Table 1: Signal power consumption of reference $S_{\mathrm{ref}}(K' = \texttt{100})$ with 2^3 tuples for κ described by 2^3 elements representing the μ, for the CPA strategy.

$\mu_0\mu_1\mu_2$	$\kappa_0\kappa_1\kappa_2$							
	000	001	010	011	100	101	110	111
000	$+1$	$+1$	-3	-1	$+1$	-1	$+3$	-1
001	$+1$	$+1$	-1	-3	-1	$+1$	-1	$+3$
010	-3	-1	$+1$	$+1$	$+3$	-1	$+1$	-1
011	-1	-3	$+1$	$+1$	-1	$+3$	-1	$+1$
100	$+1$	-1	$+3$	-1	$+1$	$+1$	-3	-1
101	-1	$+1$	-1	$+3$	$+1$	$+1$	-1	-3
110	$+3$	-1	$+1$	-1	-3	-1	$+1$	$+1$
111	-1	$+3$	-1	$+1$	-1	-3	$+1$	$+1$

the attacker aims to recover the secret K' only, but they need the additional information provided by the bits of κ as an intermediate step to reach that goal.

Degenerated cases. For $n = 3$, six out of eight tuples $S_{\mathrm{ref}_{K'}}(\kappa)$ are linearly uncorrelated for a fixed K'. The latter makes it possible to distinguish the eight hypotheses for κ with the Pearson correlation coefficient for a given K'. The only exceptions are when $K' = \texttt{000}$ and $K' = \texttt{111}$. For those values, only four reference tuples $S_{\mathrm{ref}_{K'}}(\kappa)$ are linearly uncorrelated out of the eight ones. It means that the best adversarial guess is an equally likely secret hypothesis and its bit inverse. We call those cases *degenerated*. Compared to Table 1, the tuple of signal power reference values for a degenerated case has the highest absolute values positioned on the diagonal lines. For example, let $(K' = \texttt{000}, \kappa = \texttt{010})$ be the values to guess. In the best case, the adversary obtains the solutions $(K' = \texttt{000}, \kappa = \texttt{010})$ and $(K' = \texttt{111}, \kappa = \texttt{101})$ as equally likely. When $n = 5$, the *degenerated* cases do not appear. Indeed, the tuples $S_{\mathrm{ref}_{K'}}(\kappa)$ contain more elements (2^5) because there are more possible values μ since $n = 5$. That makes it possible to distinguish all the different secret possibilities.

4.2 The Snake Attack: Combined Correlation Power Analysis

In the second strategy, the adversary combines single-bit CPA attacks to retrieve n bits of κ and then the corresponding n bits of K'. The leakage of each bit x_i is exploited with the Pearson correlation coefficients (Section 3.2). We want to reduce the number of intermediate steps for this attack to recover the 2^{2n} possible secret bits compared to the previous CPA strategy.

Our oracle is the activity function (Definition 4), which is evaluated for (K'_i, κ, μ), i.e., one bit of the targeted sequence χ_n. Thus, $m = 1$ for this strategy. To reduce the number of intermediate steps, we take a closer look at the expression of the activity function: we see that κ_i simply adds to K'_i. That means that the sum $\epsilon_i = \kappa_i \oplus K'_i$ affects the sign of the signal power consumption values. The signal reference values where $\epsilon_i = \texttt{0}$ have the opposite sign of the ones where $\epsilon_i = \texttt{1}$. Hence, the Pearson correlation coefficient $\rho_{S_{\mathrm{ref}_{K'_i}}(\kappa),P}$ has the same absolute value regardless of ϵ_i, considering that all the other parameters stay unchanged. As a result, the adversary can not determine the individual bits of the sum ϵ_i with only the signed value of the Pearson correlation coefficient. However, the adversary obtains information about the most likely hypothesis on $(\kappa_{i+1}, \kappa_{i+2})$. They then combine those bits of knowledge when targeting the following i-th bit of the sequence χ_n. The adversary thus successfully exploits the redundant information underlying the Pearson correlation coefficients about the secret bits in a divide-and-conquer manner.

As an example, for $n = 3$, acquiring the correlation coefficient values for the first, second, and third bits makes it possible to retrieve respectively (κ_1, κ_2), (κ_2, κ_0) and (κ_0, κ_1), as highlighted in Figure 3a. Furthermore, Figure 3b presents the retrieved bits of

$$
\begin{array}{ll}
i=0: & K_0' \oplus \kappa_0 \quad \boxed{\kappa_1 \qquad \kappa_2} \\
i=1: & \boxed{\kappa_0} \quad K_1' \oplus \kappa_1 \quad \boxed{\kappa_2} \\
i=2: & \boxed{\kappa_0 \qquad \kappa_1} \quad K_2' \oplus \kappa_2
\end{array}
$$

(a) $n = 3$

$$
\begin{array}{lllll}
i=0: & K_0' \oplus \kappa_0 & \boxed{\kappa_1 \qquad \kappa_2} & \kappa_3 & \kappa_4 \\
i=1: & \kappa_0 & K_1' \oplus \kappa_1 \quad \boxed{\kappa_2 \qquad \kappa_3} & & \kappa_4 \\
i=2: & \kappa_0 & \kappa_1 & K_2' \oplus \kappa_2 \quad \boxed{\kappa_3 \qquad \kappa_4} \\
i=3: & \boxed{\kappa_0} & \kappa_1 & \kappa_2 & K_3' \oplus \kappa_3 \quad \boxed{\kappa_4} \\
i=4: & \boxed{\kappa_0 \qquad \kappa_1} & \kappa_2 & \kappa_3 & K_4' \oplus \kappa_4
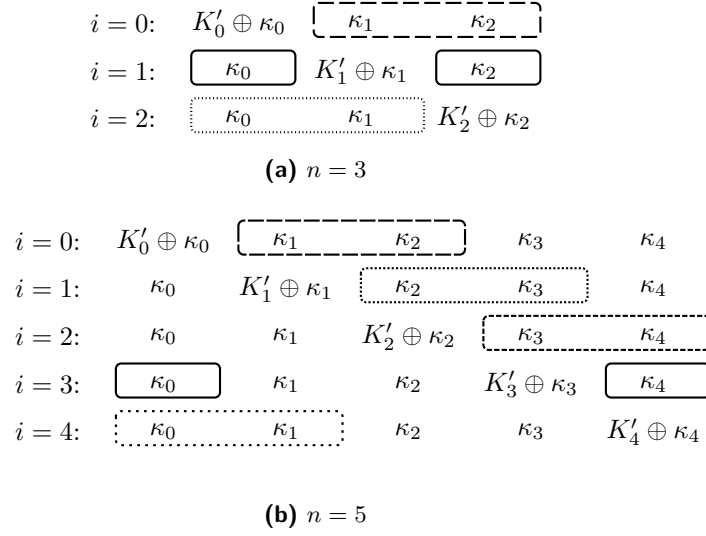\end{array}
$$

(b) $n = 5$

Figure 3: The information about the secret bits retrieved without ambiguity from the different Pearson correlation coefficients, is highlighted according to the targeted i-th bit of the sequence χ_n in different dotted lines.

κ for a sequence χ_5. The adversary thus exploits the redundant information provided by the quadratic terms of the activity function to determine the most likely κ. We henceforth denote by $*\kappa_{i+1}\kappa_{i+2}$ ($i \in \mathbb{Z}/n\mathbb{Z}$) the fact that the sign of $K_i' \oplus \kappa_i$ is discarded.

As a result, we refine the signal reference values from Definition 8 to compute the Pearson correlation coefficient. To that end, the adversary generates n *reduced* tuples of signal power consumption references S'_{ref} for each bit to combine the correlation results. To not take into account the sign effect, the adversary fixes the value of $K_i' \oplus \kappa_i$ to either 0 or 1 while generating the reduced signal reference values. We define below in a formal manner the reduced signal reference values S'_{ref} for our optimized combined CPA.

Definition 10 (Reduced signal reference values S'_{ref})**.** The reduced signal power consumption values are stored in n tuples of tuples $S'_{\text{ref}}(K_i')$, one for each targeted bit x_i of the output of S-box, which contains 2^2 tuples $S'_{\text{ref}_i}(*\kappa_{i+1}\kappa_{i+2})$, one for each value of $(\kappa_{i+1}, \kappa_{i+2})$, which consists of 2^3 signal power consumption evaluations for the meaningful values of the message μ. More formally, we have:

$$
S'_{\text{ref}} \overset{\text{def}}{=} \left(S'_{\text{ref}}(K_i') \mid i \in \mathbb{Z}/n\mathbb{Z} \right),
$$

which, for a bit i, consists of:

$$
S'_{\text{ref}}(K_i') = \left(S'_{\text{ref}_i}(*\kappa_{i+1}\kappa_{i+2}) \mid (\kappa_{i+1}, \kappa_{i+2}) \in \mathbb{F}_2^2 \right),
$$

Then, for a fixed secret value $(\kappa_{i+1}, \kappa_{i+2})$, we have:

$$
S'_{\text{ref}_i}(*\kappa_{i+1}\kappa_{i+2}) = \left(S(K_i', \kappa, \mu) \mid \mu \in \mathbb{F}_2^3 \text{ and } K_i' \oplus \kappa_i = 0 \text{ or } K_i' \oplus \kappa_i = 1 \right),
$$

where $S(K_i', \kappa, \mu)$ is the signal power consumption for $m = 1$.

Like the signal reference values S_{ref} from Definition 8 with $m = 1$, the adversary generates 2^3 values in the tuple S'_{ref_i}. That is because the algebraic expression of the activity function (Definition 4) depends on three bits of μ for a bit i.

For an explicit example, Table 2 provides the tuple $S_{\text{ref}}(K_0')$ for the first bit of the register ($m = 1$) with $n = 3$, as specified in Definition 8 for the standard CPA strategy.

We now replace it by Table 3 by discarding the tuples where $K_0 \oplus \kappa_0 = 1$ to eliminate the sign effect of ϵ_i following our *combined* CPA strategy. We have thus *reduced* signal reference values.

Table 2: Signal power consumption of reference $S_{\mathrm{ref}}(K'_0, \kappa)$. The sum $\kappa_0 \oplus K'_0$ is considered instead of κ_0.

$\mu_0\mu_1\mu_2$	$K'_0 \oplus \kappa_0 \, \kappa_1 \kappa_2$							
	000	001	010	011	100	101	110	111
000	$+1$	-1	$+1$	$+1$	-1	$+1$	-1	-1
001	-1	$+1$	$+1$	$+1$	$+1$	-1	-1	-1
010	$+1$	$+1$	$+1$	-1	-1	-1	-1	$+1$
011	$+1$	$+1$	-1	$+1$	-1	-1	$+1$	-1
100	-1	$+1$	-1	-1	$+1$	-1	$+1$	$+1$
101	$+1$	-1	-1	-1	-1	$+1$	$+1$	$+1$
110	-1	-1	-1	$+1$	$+1$	$+1$	$+1$	-1
111	-1	-1	$+1$	-1	$+1$	$+1$	-1	$+1$

Table 3: Reduced signal power consumption of reference $S'_{\mathrm{ref}}(K'_0)$. In practice and in this example, the adversary chooses to retain the tuples where $K' \oplus \kappa_0 = 0$ from Table 2 according to Definition 10.

$\mu_0\mu_1\mu_2$	$K'_0 \oplus \kappa_0 \, \kappa_1 \kappa_2$			
	*00	*01	*10	*11
000	$+1$	-1	$+1$	$+1$
001	-1	$+1$	$+1$	$+1$
010	$+1$	$+1$	$+1$	-1
011	$+1$	$+1$	-1	$+1$
100	-1	$+1$	-1	-1
101	$+1$	-1	-1	-1
110	-1	-1	-1	$+1$
111	-1	-1	$+1$	-1

In addition, we provide the reduced tuples of signal reference values for $n = 3$ and $n = 5$. For specific values of κ_{i+1} and κ_{i+2}, we write the binary sequence in monospace font with the correct amount of * to discard the unknown bits of κ (e.g., `**01*` for $n = 5$, $i = 1$, $\kappa_2 = 0$ and $\kappa_3 = 1$). We start with the reduced tuple S'_{ref} for a three-bit S-box:

$$S'_{\mathrm{ref}}(K'_0) = \left(S'_{\mathrm{ref}_0}(\texttt{*00}),\; S'_{\mathrm{ref}_0}(\texttt{*01}),\; S'_{\mathrm{ref}_0}(\texttt{*10}),\; S'_{\mathrm{ref}_0}(\texttt{*11})\right),$$
$$S'_{\mathrm{ref}}(K'_1) = \left(S'_{\mathrm{ref}_1}(\texttt{0*0}),\; S'_{\mathrm{ref}_1}(\texttt{0*1}),\; S'_{\mathrm{ref}_1}(\texttt{1*0}),\; S'_{\mathrm{ref}_1}(\texttt{1*1})\right),$$
$$S'_{\mathrm{ref}}(K'_2) = \left(S'_{\mathrm{ref}_2}(\texttt{00*}),\; S'_{\mathrm{ref}_2}(\texttt{01*}),\; S'_{\mathrm{ref}_2}(\texttt{10*}),\; S'_{\mathrm{ref}_2}(\texttt{11*})\right).$$

We can easily extend it for a five-bit S-box:

$$S'_{\mathrm{ref}}(K'_0) = \left(S'_{\mathrm{ref}_0}(\texttt{*00**}),\; S'_{\mathrm{ref}_0}(\texttt{*01**}),\; S'_{\mathrm{ref}_0}(\texttt{*10**}),\; S'_{\mathrm{ref}_0}(\texttt{*11**})\right),$$
$$S'_{\mathrm{ref}}(K'_1) = \left(S'_{\mathrm{ref}_1}(\texttt{**00*}),\; S'_{\mathrm{ref}_1}(\texttt{**01*}),\; S'_{\mathrm{ref}_1}(\texttt{**10*}),\; S'_{\mathrm{ref}_1}(\texttt{**11*})\right),$$
$$S'_{\mathrm{ref}}(K'_2) = \left(S'_{\mathrm{ref}_2}(\texttt{***00}),\; S'_{\mathrm{ref}_2}(\texttt{***01}),\; S'_{\mathrm{ref}_2}(\texttt{***10}),\; S'_{\mathrm{ref}_2}(\texttt{***11})\right),$$
$$S'_{\mathrm{ref}}(K'_3) = \left(S'_{\mathrm{ref}_3}(\texttt{0***0}),\; S'_{\mathrm{ref}_3}(\texttt{0***1}),\; S'_{\mathrm{ref}_3}(\texttt{1***0}),\; S'_{\mathrm{ref}_3}(\texttt{1***1})\right),$$
$$S'_{\mathrm{ref}}(K'_4) = \left(S'_{\mathrm{ref}_4}(\texttt{00***}),\; S'_{\mathrm{ref}_4}(\texttt{01***}),\; S'_{\mathrm{ref}_4}(\texttt{10***}),\; S'_{\mathrm{ref}_4}(\texttt{11***})\right).$$
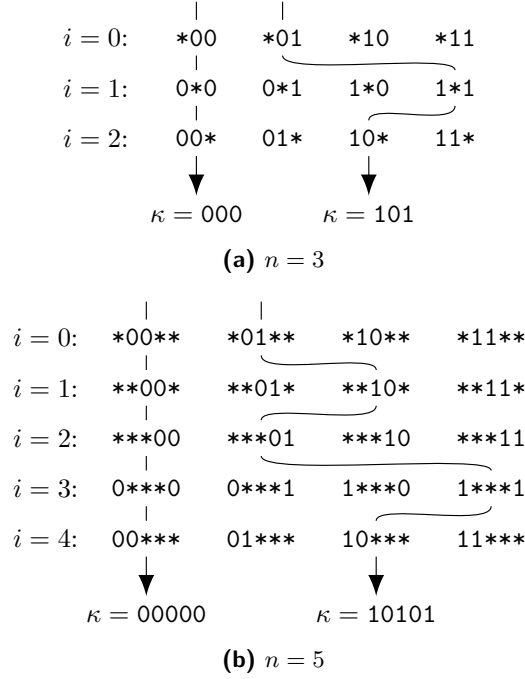
Figure 4: Combination of correlation coefficients to obtain κ values, namely the 'Snake walk'. Each label (e.g., `0***0`) represents an absolute or a squared Pearson correlation result. The values indicated in the labels are the binary sequences κ with the known bits (`0` or `1`) for κ_{i+1} and κ_{i+2}, and the unknown ones (`*`).

As a convenient result, with the reduced reference values that discard $K_i' \oplus \kappa_i$, the attacker obtains signal reference tuples $S_{\text{ref}_i}(*\kappa_{i+1}\kappa_{i+2})$ for each K_i' that have a correlation coefficient equal to zero with each other. In other words, there is only one best adversarial hypothesis on κ_{i+1} and κ_{i+2} in this strategy. Furthermore, which half of the tuples $S_{\text{ref}_i}(*\kappa_{i+1}\kappa_{i+2})$ the adversary uses is irrelevant to retrieve the bit value of κ.

The adversary then computes the n Pearson correlation coefficients $\rho_{S'_{\text{ref}_i}(*\kappa_{i+1}\kappa_{i+2}),P}$, which we henceforth write $\rho_{*\kappa_{i+1}\kappa_{i+2}}$ for readability purposes. They obtain positive and negative values to guess the secret. That is because they arbitrarily choose which four reference values to discard for each tuple due to the sign effect of ϵ_i. To eliminate the latter, the adversary squares the correlation coefficients or takes their absolute values.

The adversary then combines the absolute or squared correlation coefficients to determine the most likely hypothesis for the n bits of κ. However, selecting the highest result for each bit may provide an inconsistent value of κ due to some overlapping of the retrieved bits. To tackle this problem, the adversary chooses one correlation coefficient per bit and adds them to have coherent 2^n values of κ. For example, $\kappa = $ `011` is obtained by adding the correlation results corresponding to $\kappa = $ `*11`, $\kappa = $ `0*1`, and $\kappa = $ `01*`. There is no other combination. Figure 4a depicts how to obtain two possibilities for κ, namely $\kappa = $ `000` and $\kappa = $ `101`. For our five-bit S-box, Figure 4b presents the combination of the correlation coefficients for $\kappa = $ `00000` and $\kappa = $ `10101`. The labels in the figures symbolize the Pearson correlation coefficients $\rho_{*\kappa_{i+1}\kappa_{i+2}}$ for a particular value of $(\kappa_{i+1}, \kappa_{i+2})$ with the remaining unknown bits of κ, whose numbers depend on the length n of the considered S-box. Visually, linking the correlation results looks like performing a 'Snake walk'. Hence, the nickname of the combined CPA strategy: the *Snake attack*.

We provide in the following a definition of the Snake walk, or the combination of the Pearson correlation coefficients, to recover the bit information κ while individually

targeting the n bits of the register state. We have n reduced tuples of signal reference values $S'_{\mathrm{ref}}(K'_i)$, which contains each four reduced sub-tuples $S'_{\mathrm{ref}_i}(*\kappa_{i+1}\kappa_{i+2})$, one for each evaluation of $(\kappa_{i+1}, \kappa_{i+2})$ from Definition 10.

Definition 11 (Snake walk)**.** The Snake walk represents the addition of the n squared or absolute Pearson correlation coefficients $\rho_{*\kappa_{i+1}\kappa_{i+2}}$, such that the individual bits of κ are equal. In other words, for all bits of κ indexed by $i \in \mathbb{Z}/n\mathbb{Z}$, the value of κ_{i+2} in the index of the Pearson correlation coefficient $\rho_{*\kappa_{i+1}\kappa_{i+2}}$, which targets the i-th bit of the output of the S-box, is equal to the value of κ_{i+2} in $\rho_{*\kappa_{i+2}\kappa_{i+3}}$, which targets the $(i+1)$-th bit of the said output.

Recovering K or K'. Once a value κ is labeled as the most probable solution, the adversary can either recover K that is encoded in the entire initial register state or K' from Definition 3. To find the value K, the adversary targets all the output bits of the S-boxes stored in the register state after one round to recover all the corresponding variables κ. It results in $4n$ correlation values to manipulate times the number of parallel S-boxes for a round function. Afterward, they can determine K by applying the inverse of the linear layer λ^{-1} on the concatenation of the multiple variables κ, as done in our practical evaluations. Concretely, for one S-box, this reduces the number of correlations that need to be computed for $n = 3$ from 64 to 12 and for $n = 5$ from 1,024 to 20, resulting in a vastly more efficient attack.

Alternatively, the attacker infers the n bits of K' by retrieving the sign of the correlation values for the i-th studied bits. Indeed, the adversary can recover the initial secret by computing $\epsilon_i \oplus \kappa_i = K'_i$ for each bit x_i. That is because, from the expression of the signal power consumption, if the sign of the correlation coefficient is negative, the sum ϵ_i equals 1. Contrarily, if the sign is positive, ϵ_i equals 0. As a result, the adversary performs at most $n2^2 + n$ intermediate steps by exploiting the sign of the correlation values to recover 2^{2n} bits of information (K', κ) when targeting one S-box by taking advantage of the algebraic expression of the activity function (Definition 4). The latter heavily relies on the expression of the quadratic map of the permutation.

5   Practical Evaluation

We provide a method to determine the rank of the secret to guess. We then describe how to compute the probabilities of success for the secret bits to retrieve to be ranked at a given position. Finally, we present our hardware setup to evaluate our attack strategies.

5.1   Rank of a Secret Hypothesis

We not only intend to quantify the gain in terms of attack success probability between one attack strategy and another for a three-bit and a five-bit quadratic S-box but also to compare those probabilities for the rank of the secret solution. There are as many ranks as the number of correlations that need to be computed: it equals the number of tuples $S_{\mathrm{ref}}(K')$ times their number of sub-tuples $S_{\mathrm{ref}_{K'}}(\kappa)$ for the CPA strategy, or $4n$ for the Snake attack strategy. The secret hypotheses are ranked by their scores, which are (signed) correlation coefficients for CPA and absolute or squared coefficients for Snake.

Definition 12 (Rank of a secret hypothesis)**.** A secret hypothesis is ranked first (resp. second, etc.) if its corresponding score is the highest (resp. second highest, etc.) positive value compared to the other secret hypotheses.

The above definition is more restrictive than the order of the success rate [SMY09]. Indeed, we compute the probability to be ranked r-th, whereas the order of success rate considers whether the solution is ranked among the r-th first hypotheses.

5.2   Attack Success Probabilities for the Ranks

We aim to plot the probabilities to have the solution secret be ranked first, second, etc., for an adversary depending on their amount of power traces, also known as signal-to-noise ratio (SNR). A low SNR means manipulating a few power traces, hence much noise in the measurements. A high SNR highlights the signal part because the adversary gathered a sufficient number of power traces (Proposition 1).

We then compare the two attack strategies (Snake attack and CPA) w.r.t. the probability that an adversary can recover the solution value of the secret bits. After that, we infer which strategy performs better in terms of the required number of power traces and intermediate steps.

5.3   Hardware Setup

For our practical evaluation, we target round-based implementations of Xoodoo ($n = 3$) and Keccak-p ($n = 5$) for a width $b = 1600$, on a Spartan 6 FPGA running at $4\,\mathrm{MHz}$. We measure the current via the voltage drop over a shunt in the supply path. Then, to acquire suitable power traces, we used a ZFL-2000GH+ Low Noise Amplifier (LNA) configured with a $22\,\mathrm{dB}$ gain and a Spectrum M4 oscilloscope ($8 - \mathrm{bit}$ resolution) with a sample rate of $2.5\,\mathrm{GS/s}$.

For Xoodoo, we randomize the whole input state, adding a random key that is fixed for all measurements, before performing the first round. It should be noted that to attack Xoodoo, we also have to account for the addition of the round constant with the mapping ι and the ρ_{east} linear mapping ρ_{east} that are performed after χ_3.

In contrast to this, for Keccak, we cannot randomize the whole 1600-bit input state since the target FPGA does not have enough input pins. We tackle this by randomizing only 128 bits and measuring 13 rounds. Following this, we target the transition from round 12 to 13. The reason for this is that after 12 rounds, we are sure that the whole state is randomized and view the state at this point as being a randomized message M added to a fixed key K. Thus, we choose the random key K after the measurement and compute the fictional M from the randomized state. Then we attack as known and we also compute $\mu = \lambda(M)$ and account for the addition of the round constant with the step ι.

We have implemented both the standard CPA attack and the Snake attack in C, making use of multi-threading. Furthermore, we will make our implementations publicly available upon acceptance of the paper.

6   Attack Success Probabilities Results

First, we present our practical results for Keccak-p and Xoodoo with the two attack strategies. Secondly, we argue about the probability distribution obtained with the Henery model.

6.1   Practical Results

We present discrete attack success probabilities computed from measurements from the hardware testbed in Figures 5a and 5b for Xoodoo and Figures 6b and 6a for Keccak-p. The probability of being ranked first is depicted in black. The gray curves represent the probability of being ranked second, third, etc. For those curves, the higher the rank (rank two being the highest), the slower they collapse. In other words, the higher the rank for the gray curves, the higher their relative position is in the figures. In addition, the success probability to be ranked first equals one for a high SNR (i.e., a high amount of power traces), whereas the other ranks have a zero probability. Note that to make the gray curves more visible, we limit the x-axis to 25,000 traces for Xoodoo and 400,000 traces for
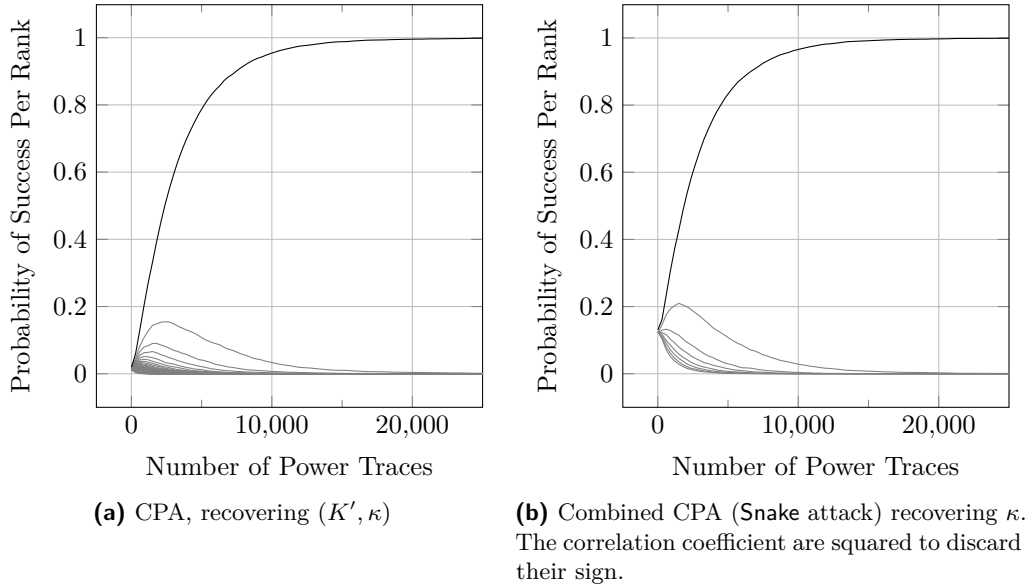
(a) CPA, recovering (K', κ)

(b) Combined CPA (Snake attack) recovering κ. The correlation coefficient are squared to discard their sign.

Figure 5: Ranked success probabilities targeting one sequence χ_3.

(a) CPA, recovering (K', κ)

(b) Combined CPA (Snake attack) recovering κ. The correlation coefficients are squared to discard their sign.
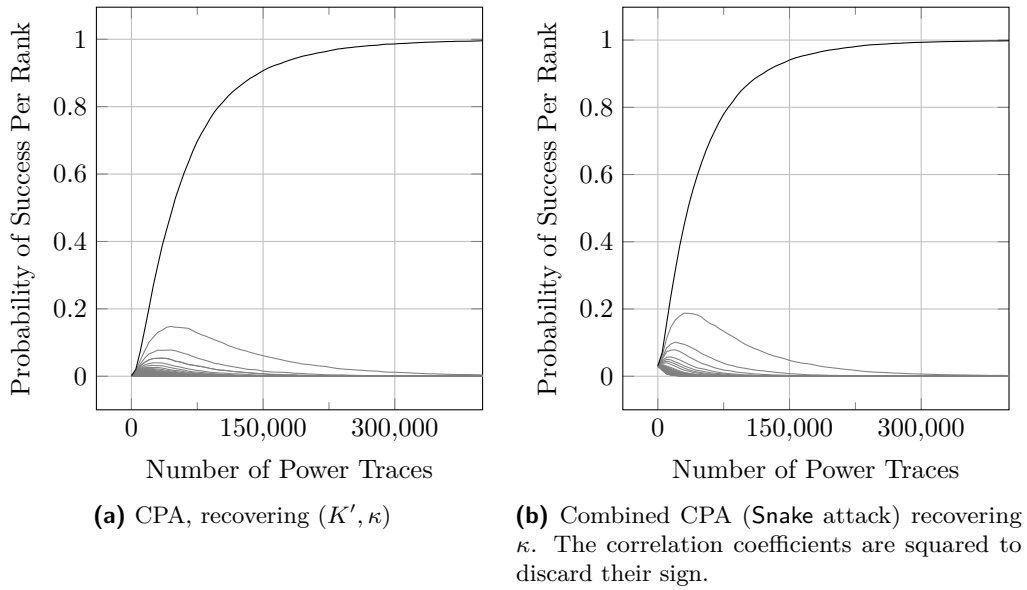
Figure 6: Ranked success probabilities targeting one sequence χ_5.

Keccak-p, respectively, even though the first rank (black curve) has not yet reached the probability of being one at this point.

For Xoodoo, the results were generated for sets of power traces that contain 75,000 traces each. The attacks were performed 16,896 times each for CPA and combined CPA/Snake attack. The initial success probabilities (i.e., full noise or low SNR) are close to be equally distributed for each strategy: ideally, they are equal to $1/2^{2n}$ for CPA and $1/2^n$ for the Snake attack. However, in practice, the first probability already includes a low number of processed traces, which leads to a slight deviation from the ideal value.

Due to the much larger state of Keccak-p compared to Xoodoo, there is a lot more noise when targeting the nonlinear layer of Keccak-p. It is reflected in Figures 6b and 6a by the

(a) $n = 3$ **(b)** $n = 5$
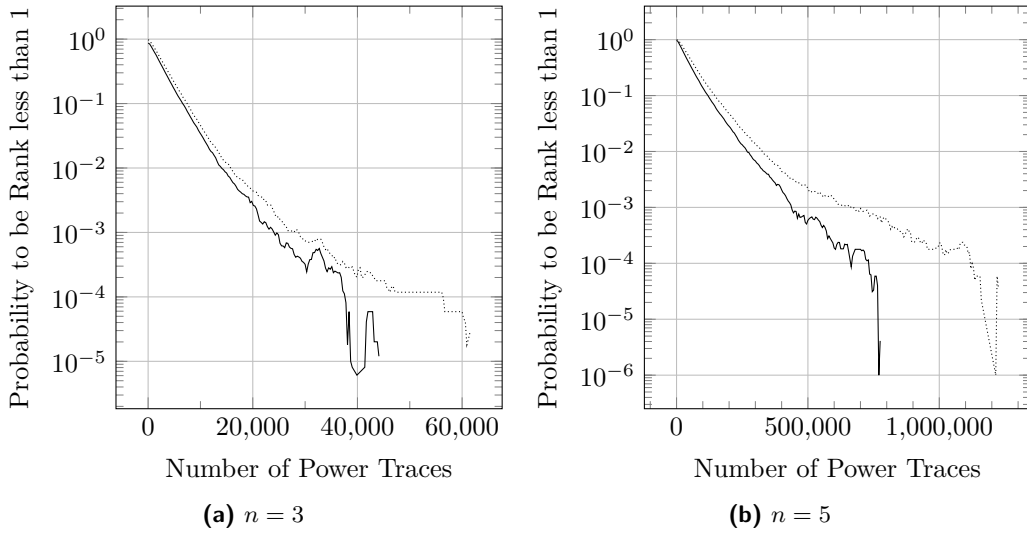
Figure 7: Comparison between CPA (dotted line) and Snake attack (solid line) for the probability of the correct hypothesis to *not* be rank one.

increased number of traces needed to recover the correct secret. Indeed, we have 1.5 million traces per set. To match the Xoodoo experiment, we conducted 16,896 experiments.

As a result, the Snake attack on Xoodoo recovers the correct κ after 43,860 power traces for all experiments. The standard CPA attack takes 61,380 traces until all experiments label the solution (K', κ) as the first rank. We also recall that for each Snake attack experiment, only 12 correlations must be computed, compared to 64 for standard CPA.

For Keccak, there are even higher differences concerning the number of power traces. Indeed, the Snake attack consistently finds the value of the solution κ after 771,600 traces for all experiments. In contrast, the standard CPA attack requires 1,223,400 traces. Computationally, the Keccak Snake attack requires computing 20 correlations: it outperforms the standard CPA attack, which operates on 1,024 correlations.

Figures 7a and 7b directly compare the probabilities that the solution hypothesis is *not* ranked first. We can see that the Snake attack constantly has a lower probability than the standard CPA approach, demonstrating that the Snake attack outperforms the standard CPA approach. In summary, the quadratic function χ_n is attacked with fewer efforts thanks to taking advantage of its algebraic expression with combined CPA, or Snake attack, compared to the standard CPA.

6.2 Probability Distribution: Henery Model

If we consider our approach from a probabilistic modeling perspective, we are comparing and ranking 2^{2n}, or 2^n, possible outcomes in the form of Pearson correlation coefficients between observed and reference power consumption values. The probability of success for each outcome (i.e., secret hypothesis) is given by how close it is to the original solution. We note that the probabilities of success in Figures 6a and 5a add up to one for a given number of traces. They are dependent of each other and follow a particular order (the higher the probability of success, the higher the rank). It is akin to ranking the outcomes in a horse race according to the horses' probabilities of winning. This type of problem was studied and formalized by Henery in 1981 [Hen81].

In the latter, the author defines a probability model over each outcome permutation (i.e., ranking in this work) given a particular distribution family for the outcome variables.

Our ranked success probabilities follow a Henery model, but it is unfortunately (generally) impossible to derive an analytically tractable expression for them, given an arbitrarily distributed outcome. Thus, we rely instead on real-world measurements.

7   Discussion

In this section, we refer to existing attacks in the literature that employ similar side-channel methods to the ones presented in this paper. We discuss the similarities and differences with our work in this paper.

Samwel and Daemen [SD17] extended the work of Bertoni *et al.* [BDD+12] by targeting the full sequence χ_5 as implemented in Lake Keyak, with DPA on a FPGA instead of targeting only one bit of the said S-box. It is similar to our standard CPA attack scenario from Section 4.1. They concluded that attacking a full binary sequence is less efficient than attacking a single bit of the non-linear layer. We continue this work by providing more detailed insight into the attack on the full sequence χ_5 with CPA and by formalizing the attack in order to generalize it to any χ_n. Indeed, we provide the probabilities of success according to the ranks and detail the tuples of signal reference values. We also study the case for χ_3 and exhibit the potential degenerated cases that an adversary might encounter. We improve on the single-bit attack with Snake, or combined CPA, by effectively reducing the number of intermediate results to compute.

Furthermore, Messerges *et al.* presented the multi-bit DPA approach [MDS02] where, instead of considering the leakage of one bit with the selection function, the attacker considers d bits. From there, three partitions are created depending on the values of the output of the selection function for d bits: one where the output of the selection function is 0s, a second one for all 1s, and a third one for mixed values. The meaning of partitions in this work corresponds to the signal reference values in our case. However, our partitions—tuples of signal reference values—reflect the precise number of bit flips.

The latter work inspired the authors from [LCC+06], which proposed the Partitioning Power Analysis (PPA) as a more general method to exploit the power consumption signals of d bits instead of only one. Their presented attack is similar to CPA but without normalizing the correlation coefficient by the standard deviation of the power consumption values. They classify power traces into $(d+1)$ classes according to the Hamming distance of the considered d bits between a previous and current state. This approach is included in our work for the classic CPA approach, where the signal reference values can have values ranging from $-n$ to $+n$ depending on the Hamming distance of the state before and after one round. We further extend their work by combining the Pearson correlation coefficients to have coherent secret values, as in the Snake attack.

Other works in power analysis have similar notions to our tuple of signal reference values. Another example is the power analysis attack presented by Standaert *et al.* for an FPGA implementation of AES [SÖP04]. They used a *selected prediction matrix* similar to our tuples of signal reference values. Nevertheless, they do not link the reference values among themselves to reduce the complexity of their side-channel attack unlike our work.

On the other hand, combining leakage signals is used for higher-order masking schemes [CPR07], where *masking* refers to splitting a secret variable into several *shares*. A power analysis attack is applied to the individual shares of the secret. It uses a combining function (e.g., a product, a difference of absolute values) on them to find a high correlation with the original un-shared value. However, the term *combination* is used differently than in our paper for combined CPA, for which we exploit the information redundancy to reduce the number of intermediate steps to compute.

Some attacks were proposed to reduce the complexity of the power analysis, like the Butterfly-Attack [ZKS12] presented by Zohner *et al.* for the modular addition. It comes in handy to reduce the complexity of a DPA attack on this operand by lowering the

number of hypothesis computations in a divide-and-conquer manner and then exploiting the information on the symmetric counterpart of the solution key. They exploit the latter because, by splitting the modular addition results into blocks, the divide-and-conquer results no longer highlight the solution key with the highest correlation value. Nonetheless, it is applied to a different kind of cipher (that uses modular addition) and in a different context than the permutations for symmetric cryptography targeted in this paper.

Applicability to Ascon. The NIST Lightweight Cryptography competition winner, Ascon, has an S-box that is affine equivalent to χ_5 [DEMS21]. It operates on each bit-slice of five registers that contain 64 bit-word and are denoted by $x_{i,0 \leq i \leq 4}$. For the first round of the first permutation, the register x_0 is initialized by a constant C (or Initialization Vector), x_1 by a first key word K_1, x_2 by a second key word K_2, x_3 by a first nonce word N_1 and x_4 by a second nonce word N_2. For the same bit position for the 64 bit-word, we can write the Algebraic Normal Form (ANF) of the S-box after its first execution (operations carried in $\mathbb{F}_2$):

$$x_0 = C \oplus N_1 \oplus K_2 \oplus K_1(C \oplus N_2 \oplus K_2 \oplus 1)\,,$$
$$x_1 = C \oplus N_1 \oplus N_2 \oplus K_1(N_1 \oplus K_2 \oplus 1) \oplus K_2(N_1 \oplus 1)\,,$$
$$x_2 = N_2 \oplus N_1 N_2 \oplus K_1 \oplus K_2 \oplus 1\,,$$
$$x_3 = C \oplus N_1 \oplus N_2 \oplus C(N_1 \oplus N_2) \oplus K_1 \oplus K_2\,,$$
$$x_4 = N_1 \oplus N_2 \oplus K_1(C \oplus N_2 \oplus 1)\,.$$

The Snake attack exploits the redundancy of quadratic terms, where some key bits appear. It also assumes that there are as many key bits as the size of the S-box. We observe, based on the above equations, that the first call of the Ascon S-box does not fulfill those requirements: there are only two key bits for a five-bit S-box and only the registers x_0 and x_1 contain quadratic terms with two key bits. In that case, the Snake attack does not provide any advantage, and we refer to Samwel and Daemen [SD17] for a detailed DPA attack on the Ascon S-box with the same leakage model as ours.

8 Future Work

In this paper, we presented in details CPA attacks on χ_3 (Xoodoo) and χ_5 (Keccak-*p*) to understand in depth how to optimize standard correlation power analysis. They can be extended to larger quadratic S-boxes employed in other ciphers, for which the difference in the number of intermediate results will be even more drastic between the Snake attack and the standard CPA. For example, one could target Subterranean 2.0 [DMMR20] NIST Lightweight Cryptography competition candidate that uses as an S-box the function χ_{257}, or Rasta [DEG+18] and Dasta [HL20] that use χ_n with n being the block length and an odd number. Furthermore, to apply our Snake attack to different quadratic functions (other than χ), one would need to change the activity function accordingly, then compute the corresponding signal reference vectors and sum the correlation results according to our Snake walk.

Additionally, our Snake attack heuristically combines multiple one-bit CPA attacks, which opens the door for an approach that relies on DPA instead since only two classes are to be distinguished. An ad-hoc porting of the Snake attack could include the computation of Welch's t-test in place of CPA, and then the t-test results would be heuristically combined with the 'Snake walk' to obtain the secret κ. We performed preliminary experiments on the same trace sets of Xoodoo used for the results shown in Figure 5b. However, we observed that the sum of squared t values results in a weaker attack that requires more traces than the standard CPA approach. Thus, a method to rely on DPA remains for future work.

Moreover, it would be interesting to assess whether our method can be applied to other types of leakage, e.g., electromagnetic information with Differential Electromagnetic Analysis (DEMA) introduced by Quisquater and Samyde [QS01].

To extend the scope of this study, one could try to recover the initial secret after several rounds or a masked implementation with the help of the presented model and theory. As the Snake attack exploits the information redundancy of quadratic terms to find the secret values, it seems a natural fit for masked hardware implementations of cryptographic algorithms (like Threshold Implementations [NRR06]), which are designed to split the secret bits into several variables each.

9   Conclusion

We considered the difference in power consumption of register cells before and after one round of a hypothetical implementation of a permutation-based algorithm for quadratic maps, which are used as S-boxes. This work details the case for χ_n with $n \in \{3, 5\}$. We model the flipping activity of the register state with simple assumptions, which takes advantage of the cryptanalysis of quadratic maps. We correlate the measured consumed power for the targeted bits with reference values by the mean of the Pearson correlation coefficient. Afterward, we study the success probabilities of CPA and combined CPA, or Snake attack, to retrieve up to $2n$ secret bits. To the best of our knowledge, we are the first to describe both the standard CPA attack and the novel combined CPA for quadratic maps.

We demonstrate that an adversary performs better with our novel combined CPA strategy (Snake attack) than the standard CPA approach for small S-boxes. In order words, the attack uses $n2^2 + n$ intermediate steps for the combined attack, instead of 2^{2n} for the classic one, to determine 2^{2n} secret bits. Not only we successfully reduce the computational complexity of the standard CPA attack with the Snake attack, but we also require fewer power traces to find the secret solution. In addition, for the CPA strategy for Xoodoo, the adversary can recover all secret bits but encounters two degenerated cases that give two equally likely secret hypotheses. We also propose a probability distribution model for ranked attack success probabilities. Our study highlights that a tailored CPA strategy that uses the cryptanalysis of a quadratic map performs better than traditional CPA.

Acknowledgement

References

[ABP+18]   Victor Arribas, Begül Bilgin, George Petrides, Svetla Nikova, and Vincent Rijmen. Rhythmic keccak: SCA security and low latency in HW. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2018(1):269–290, 2018.

[BBS17]   Dusan Bozilov, Begül Bilgin, and Haci Ali Sahin. A note on 5-bit quadratic permutations' classification. *IACR Trans. Symmetric Cryptol.*, 2017(1):398–404, 2017.

[BCO04]   Eric Brier, Christophe Clavier, and Francis Olivier. Correlation power analysis with a leakage model. In *CHES*, volume 3156 of *Lecture Notes in Computer Science*, pages 16–29. Springer, 2004.

[BDD+12]   Guido Bertoni, Joan Daemen, Nicolas Debande, Thanh-Ha Le, Michaël Peeters, and Gilles Van Assche. Power analysis of hardware implementations protected with secret sharing. In *MICRO Workshops*, pages 9–16. IEEE Computer Society, 2012.

[BDP+16a] Guido Bertoni, Joan Daemen, Michaël Peeters, Gilles Van Assche, and Ronny Van Keer. Caesar submission: Ketje v2. *Submission to CAESAR*, 2016.

[BDP+16b] Guido Bertoni, Joan Daemen, Michaël Peeters, Gilles Van Assche, and Ronny Van Keer. Caesar submission: Keyak v2. *Submission to CAESAR*, 2016.

[BDPV08]   Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. On the indifferentiability of the sponge construction. In *EUROCRYPT*, volume 4965 of *Lecture Notes in Computer Science*, pages 181–197. Springer, 2008.

[BDPV11]   Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. The keccak reference. *Submission to NIST (Round 3)*, 13:14–15, 2011.

[BDPVA10] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Building power analysis resistant implementations of Keccak. In *Second SHA-3 candidate conference*, volume 142. Citeseer, 2010.

[BNN+15]   Begül Bilgin, Svetla Nikova, Ventzislav Nikov, Vincent Rijmen, Natalia N. Tokareva, and Valeriya Vitkup. Threshold implementations of small S-boxes. *Cryptogr. Commun.*, 7(1):3–33, 2015.

[CPR07]   Jean-Sébastien Coron, Emmanuel Prouff, and Matthieu Rivain. Side channel cryptanalysis of a higher order masking scheme. In *CHES*, volume 4727 of *Lecture Notes in Computer Science*, pages 28–44. Springer, 2007.

[Dae95]   Joan Daemen. *Cipher and hash function design strategies based on linear and differential cryptanalysis.* PhD thesis, Doctoral Dissertation, March 1995, KU Leuven, 1995.

[DC07]   Christophe De Cannière. Analysis and design of symmetric encryption algorithms. *Doctoral Dissertaion, KULeuven*, 2007.

[DEG+18]   Christoph Dobraunig, Maria Eichlseder, Lorenzo Grassi, Virginie Lallemand, Gregor Leander, Eik List, Florian Mendel, and Christian Rechberger. Rasta: A cipher with low ANDdepth and few ANDs per bit. In *CRYPTO (1)*, volume 10991 of *Lecture Notes in Computer Science*, pages 662–692. Springer, 2018.

[DEMS21]   Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. Ascon v1.2: Lightweight authenticated encryption and hashing. *J. Cryptol.*, 34(3):33, 2021.

[DHP+20]   Joan Daemen, Seth Hoffert, Michaël Peeters, Gilles Van Assche, and Ronny Van Keer. Xoodyak, a lightweight cryptographic scheme. *IACR Trans. Symmetric Cryptol.*, 2020(S1):60–87, 2020.

[DHVV18]   Joan Daemen, Seth Hoffert, Gilles Van Assche, and Ronny Van Keer. The design of Xoodoo and Xoofff. *IACR Trans. Symmetric Cryptol.*, 2018(4):1–38, 2018.

[DMM21]   Joan Daemen, Alireza Mehrdad, and Silvia Mella. Computing the distribution of differentials over the non-linear mapping χ. In *SPACE*, volume 13162 of *Lecture Notes in Computer Science*, pages 3–21. Springer, 2021.

[DMMR20]   Joan Daemen, Pedro Maat Costa Massolino, Alireza Mehrdad, and Yann Rotella. The subterranean 2.0 cipher suite. *IACR Trans. Symmetric Cryptol.*, 2020(S1):262–294, 2020.

[Dwo15]   Morris Dworkin. SHA-3 standard: Permutation-based hash and extendable-output functions, 08 2015.

[EME22]   Johannes Erlacher, Florian Mendel, and Maria Eichlseder. Bounds for the security of Ascon against differential and linear cryptanalysis. *IACR Trans. Symmetric Cryptol.*, 2022(1):64–87, 2022.

[GPT21]   David Gérault, Thomas Peyrin, and Quan Quan Tan. Exploring differential-based distinguishers and forgeries for ASCON. *IACR Trans. Symmetric Cryptol.*, 2021(3):102–136, 2021.

[Hen81]   Robert J. Henery. Permutation probabilities as models for horse races. *Journal of the royal statistical society series b-methodological*, 43:86–91, 1981.

[HL20]   Phil Hebborn and Gregor Leander. Dasta - alternative linear layer for Rasta. *IACR Trans. Symmetric Cryptol.*, 2020(3):46–86, 2020.

[KJJ99]   Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *CRYPTO*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999.

[LCC+06]   Thanh-Ha Le, Jessy Clédière, Cécile Canovas, Bruno Robisson, Christine Servière, and Jean-Louis Lacoume. A proposition for correlation power analysis enhancement. In *CHES*, volume 4249 of *Lecture Notes in Computer Science*, pages 174–186. Springer, 2006.

[LIMY20]   Fukang Liu, Takanori Isobe, Willi Meier, and Zhonghao Yang. Algebraic attacks on round-reduced Keccak/Xoodoo. *IACR Cryptol. ePrint Arch.*, page 346, 2020.

[LL02]   Don S. Lemons and Paul Langevin. *An introduction to stochastic processes in physics.* 2002.

[LP07]   Gregor Leander and Axel Poschmann. On the classification of 4 bit S-boxes. In *WAIFI*, volume 4547 of *Lecture Notes in Computer Science*, pages 159–176. Springer, 2007.

[LQT19]   Guozhen Liu, Weidong Qiu, and Yi Tu. New techniques for searching differential trails in Keccak. *IACR Trans. Symmetric Cryptol.*, 2019(4):407–437, 2019.

[LSL21]   Yunwen Liu, Siwei Sun, and Chao Li. Rotational cryptanalysis from a differential-linear perspective - practical distinguishers for round-reduced FRIET, Xoodoo, and Alzette. In *EUROCRYPT (1)*, volume 12696 of *Lecture Notes in Computer Science*, pages 741–770. Springer, 2021.

[MB19]   Lauren De Meyer and Begül Bilgin. Classification of balanced quadratic functions. *IACR Trans. Symmetric Cryptol.*, 2019(2):169–192, 2019.

[MDS02]   Thomas S. Messerges, Ezzy A. Dabbish, and Robert H. Sloan. Examining smart-card security under the threat of power analysis attacks. *IEEE Trans. Computers*, 51(5):541–552, 2002.

[MOS11]   Stefan Mangard, Elisabeth Oswald, and François-Xavier Standaert. One for all - all for one: unifying standard differential power analysis attacks. *IET Information Security*, 5(2):100–110, 2011.

[MPLJ19]   Luca Mariot, Stjepan Picek, Alberto Leporati, and Domagoj Jakobovic. Cellular automata based S-boxes. *Cryptogr. Commun.*, 11(1):41–62, 2019.

[NRR06]   Svetla Nikova, Christian Rechberger, and Vincent Rijmen. Threshold implementations against side-channel attacks and glitches. In *ICICS*, volume 4307 of *Lecture Notes in Computer Science*, pages 529–545. Springer, 2006.

[PAM19]   Antoon Purnal, Victor Arribas, and Lauren De Meyer. Trade-offs in protecting Keccak against combined side-channel and fault attacks. In *COSADE*, volume 11421 of *Lecture Notes in Computer Science*, pages 285–302. Springer, 2019.

[PSDQ05]   Eric Peeters, François-Xavier Standaert, Nicolas Donckers, and Jean-Jacques Quisquater. Improved higher-order side-channel attacks with FPGA experiments. In *CHES*, volume 3659 of *Lecture Notes in Computer Science*, pages 309–323. Springer, 2005.

[QS01]   Jean-Jacques Quisquater and David Samyde. Electromagnetic analysis (EMA): measures and counter-measures for smart cards. In *E-smart*, volume 2140 of *Lecture Notes in Computer Science*, pages 200–210. Springer, 2001.

[SD17]   Niels Samwel and Joan Daemen. DPA on hardware implementations of Ascon and Keyak. In *Conf. Computing Frontiers*, pages 415–424. ACM, 2017.

[SD23]   Jan Schoone and Joan Daemen. The state diagram of χ. Cryptology ePrint Archive, Paper 2023/328, 2023.

[SMY09]   François-Xavier Standaert, Tal Malkin, and Moti Yung. A unified framework for the analysis of side-channel key recovery attacks. In *EUROCRYPT*, volume 5479 of *Lecture Notes in Computer Science*, pages 443–461. Springer, 2009.

[SÖP04]   François-Xavier Standaert, Siddika Berna Örs, and Bart Preneel. Power analysis of an FPGA: implementation of rijndael: Is pipelining a DPA countermeasure? In *CHES*, volume 3156 of *Lecture Notes in Computer Science*, pages 30–44. Springer, 2004.

[ZBRL15]   Wentao Zhang, Zhenzhen Bao, Vincent Rijmen, and Meicheng Liu. A new classification of 4-bit optimal s-boxes and its application to PRESENT, RECTANGLE and SPONGENT. In *FSE*, volume 9054 of *Lecture Notes in Computer Science*, pages 494–515. Springer, 2015.

[ZKS12]   Michael Zohner, Michael Kasper, and Marc Stöttinger. Butterfly-attack on Skein's modular addition. In *COSADE*, volume 7275 of *Lecture Notes in Computer Science*, pages 215–230. Springer, 2012.

[ZSS⁺21] Sara Zarei, Aein Rezaei Shahmirzadi, Hadi Soleimany, Raziyeh Salarifard, and Amir Moradi. Low-latency Keccak at any arbitrary order. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2021(4):388–411, 2021.