Appears in *Cryptographic Hardware and Embedded Systems—CHES 2005*, *Lecture Notes in Computer Science* **3659 (2005) 293–308. Springer-Verlag.**

On Second-Order Differential Power Analysis*

Marc Joye[1,**], Pascal Paillier[2], and Berry Schoenmakers[3]

[1] CIM-PACA, Centre de Micro-électronique de Provence – George Charpak
Avenue des Anénomes, Quartier Saint Pierre, 13120 Gardanne, France
marc.joye@gemplus.com
[2] Advanced Research and Security Centre, Gemplus S.A.
34 rue Guynemer, 92447 Issy-les-Moulineaux, France
pascal.paillier@gemplus.com
[3] Dept of Mathematics and Computing Science, Eindhoven University of Technology
P.O. Box 513, 5600 MB Eindhoven, The Nederlands
berry@win.tue.nl

Abstract. Differential Power Analysis (DPA) is a powerful cryptanalytic technique aiming at extracting secret data from a cryptographic device by collecting power consumption traces and averaging over a series of acquisitions. In order to prevent the leakage, hardware designers and software programmers make use of masking techniques (a.k.a. data whitening methods). However, the resulting implementations may still succumb to second-order DPA. Several recent papers studied second-order DPA but, although the conclusions that are drawn are correct, the analysis is not.
This paper fills the gap by providing an exact analysis of second-order DPA as introduced by Messerges. It also considers several generalizations, including an extended analysis in the more general Hamming-distance model.

Keywords. Side-channel analysis, differential power analysis, second-order attacks.

1 Introduction

Undoubtedly, power analysis attacks constitute a cheap yet powerful cryptanalytic approach for extracting secret data from smart cards or other embedded crypto-enabled devices. Among them, Differential Power Analysis (DPA) as suggested in [8] presents the practical advantage of allowing data extraction even though the attacker has only a weak knowledge of the device being attacked. However, the original statistical technique behind DPA may require the acquisition of many power traces to average away random and computational noises.

* The work described in this paper has been supported in part by the European Commission through the IST Programme under Contract IST-2002-507932 ECRYPT.
** Seconded from Gemplus.

Many first-order variations of DPA, as well as other approaches such as direct correlation (e.g., [5]), have emerged since [8] that lead to performance improvements by lowering the number of recorded power traces.

The commonly suggested way to thwart first-order power analysis is random masking [6] a.k.a *data whitening* wherein intermediate computations are handled under a probabilistic form to defeat statistical correlation. In fact, Boolean and arithmetic maskings are certainly the most intensively used approach to protect power-sensitive cryptographic software as it appears that data randomization usually works well in practice, even when hardware countermeasures are not available. It is known, however, that masking can be defeated if the attacker knows how to correlate power consumption more than once per computation. This is known as *second-order*, or more generally *higher-order*, power analysis and was originally suggested by Messerges in [12]. These attacks are known to be more complex and delicate to carry out because they usually require the attacker to have a deeper knowledge of the device, although this might be alleviated in particular cases [16]. Investigating second-order power attacks, however, is of major importance for practitioners as it remains the only known way that is powerful enough to break real-life, DPA-protected security products.

Amazingly, second-order power analysis has remained essentially empirical so far and has never been investigated *analytically*, i.e., by the means of a direct mathematical reasoning. Second-order attacks are often described in a specific statistical setting that relies on first-order DPA one way or another, thereby leaving expected observations vague and cost estimations without a clear statement. This paper adopts a totally different approach. We formally *compute* what one expects from second-order attacking any randomized algorithm and express the amplitude of observed peaks as a function of hardware-dependent parameters. Although we essentially consider the case of Boolean masking, our results extend in several directions.

The rest of this paper is organized as follows. The next section reviews the concept of power analysis, including SPA, DPA and its higher-order generalizations. Section 3 is the core of our paper. We explain why and when second-order DPA works and carefully evaluate the height of the expected DPA peak. Next, in Section 4 we extend our main result from the Hamming-weight model to the Hamming-distance model. An experimental validation on a 1st-order protected implementation of RC6 is provided in Section 5. Finally, we conclude in Section 6.

2 Power Analysis

The power consumption of a (cryptographic) device can be monitored with an oscilloscope by inserting a resistor between the ground or Vcc pins and the actual ground. As the power consumption may vary depending on the manipulated data, some secret information may leak. This is the basic idea behind power analysis, and differential power analysis [8] in particular. In addition to power consumption, other side channels have been considered, including timing [11] and electromagnetic radiation (EM) [7, 14, 3].

2.1 SPA attacks

In simple power analysis (SPA) attacks, an adversary tries to relate the power consumption to the data being handled from essentially a *single* power consumption trace (which may in turn be obtained as the average trace for a number of traces corresponding to identical data, to reduce the noise level).

To be successful in this kind of attack, however, the adversary should have (or get) the knowledge of implementation details of the system being attacked.

2.2 DPA attacks

Differential power analysis (DPA) attacks are more powerful due to their generic nature. In this kind of attack, an adversary collects several power consumption traces for different inputs and applies statistical techniques to retrieve secret information.

As an illustration, consider the following example. Suppose that at some point, an intermediate value, say $\mathtt{I}(x, s)$, only depends on known data x and on a small portion of secret data s (i.e., small enough so that all possible values for s can be exhausted). Then for each possible value $\hat{s}$ for s, the attacker prepares two sets, $\mathfrak{S}_0(\hat{s})$ and $\mathfrak{S}_1(\hat{s})$, defined as:

$$\mathfrak{S}_b(\hat{s}) = \big\{x \mid g(\mathtt{I}(x, \hat{s})) = b\big\} \quad \text{for } b \in \{0, 1\} \tag{1}$$

where g is an appropriate *Boolean selection function* (see later). The next step consists in averaging the corresponding power consumption traces. With $\langle\cdot\rangle$ denoting the average operator and $\mathscr{C}(t)$ denoting the power consumption of the device under analysis at time period t, the adversary evaluates the *(first-order) DPA trace*

$$\Delta_1(\hat{s}, t) = \langle\mathscr{C}(t)\rangle_{x\in\mathfrak{S}_1(\hat{s})} - \langle\mathscr{C}(t)\rangle_{x\in\mathfrak{S}_0(\hat{s})} \ . \tag{2}$$

Hence, $\Delta_1(\hat{s}, t)$ is the difference of the average power consumption for sets $\mathfrak{S}_1(\hat{s})$ and $\mathfrak{S}_0(\hat{s})$, for each time period t. Assuming that (i) the intermediate data $\mathtt{I}(x, s)$ always occurs at the same time period $t = \tau$, and (ii) there are sufficiently many values for x so that $\mathtt{I}(x, \hat{s})$ is close to the uniform distribution, the DPA trace $\Delta_1(\hat{s}, t)$ exhibiting the highest peak (at time period τ) is likely the one for which $\hat{s} = s$. This way, the adversary recovers the value of secret data s.

Why does this work? Basically, the purpose of DPA is to magnify the effect of a single bit within a machine word. Suppose that a random word in a Ω-bit processor is known. Suppose further that the associated power consumption obeys the Hamming-weight model, which means that power variations are correlated to the Hamming weight (i.e., number of non-zero bits) of the manipulated data words. If the selection function $g(w)$ used to construct the sets $\mathfrak{S}_0(\hat{s})$ and $\mathfrak{S}_1(\hat{s})$ (see Eq. (1)) returns the value of a bit in the representation of word w, it follows that the words $\mathtt{I}(x, \hat{s})$ of set $\mathfrak{S}_0(\hat{s})$ have an average Hamming weight of $(\Omega - 1)/2$ whereas the words $\mathtt{I}(x, \hat{s})$ of set $\mathfrak{S}_1(\hat{s})$ have an average Hamming weight of $(\Omega + 1)/2$. As a result, the DPA trace has the property of causing a DPA peak when the selection bit, $g(\mathtt{I}(x, s))$, is handled.

2.3 Higher-order DPA attacks

k^{th}-order DPA attacks generalize (first-order) DPA attacks by considering simultaneously k samples —within the same power consumption trace— that correspond to k different intermediate values.

The main application of higher-order DPA attacks is to attack systems protected against first-order DPA [12]. A method commonly used to thwart (first-order) DPA attacks is the so-called *data whitening method*. Each intermediate sensitive data is XOR-ed with a random value, unknown to the adversary. Back to our illustration of Section 2.2, this means that the value of $w = \mathtt{I}(x, s)$ is XOR-ed with a random value r. Therefore, the adversary sees no longer a DPA peak in $\Delta_1(s, t)$ (cf. Eq. (2)) when $t = \tau$ and the attack fails.

However, if the adversary knows the time periods, τ_1 and τ_2 ($\tau_1 \neq \tau_2$), when the values of r and of $w \oplus r$ are manipulated, respectively, then she can evaluate

$$\overline{\Delta_2}(\hat{s}) = \left\langle |\mathscr{C}(\tau_2) - \mathscr{C}(\tau_1)| \right\rangle_{x \in \mathfrak{S}_1(\hat{s})} - \left\langle |\mathscr{C}(\tau_2) - \mathscr{C}(\tau_1)| \right\rangle_{x \in \mathfrak{S}_0(\hat{s})} \ . \tag{3}$$

The value $\hat{s}$ for which $\overline{\Delta_2}(\hat{s})$ is maximal (in absolute value) is likely $\hat{s} = s$ and again the adversary recovers the value of secret data s [12].

In case the adversary only knows the offset $\delta = \tau_2 - \tau_1$ (but not τ_1 nor τ_2), the previous attack can be extended as follows (cf. "known-offset 2DPA" of [16]). The adversary evaluates the *second-order DPA trace*

$$\Delta_2(\hat{s}, t) = \left\langle |\mathscr{C}(t + \delta) - \mathscr{C}(t)| \right\rangle_{x \in \mathfrak{S}_1(\hat{s})} - \left\langle |\mathscr{C}(t + \delta) - \mathscr{C}(t)| \right\rangle_{x \in \mathfrak{S}_0(\hat{s})} \ .$$

Again, under certain assumptions, the second-order DPA trace exhibiting the highest DPA peak will likely uncover the value of s.

3 Evaluating Second-Order DPA Peaks

3.1 Basic result

Let $\mathsf{H}(x)$ denote the Hamming weight of x. A simple model for power leakage is the *(generalized) Hamming-weight model*. This model assumes that the (instantaneous) power consumption $\mathscr{C}$ is linearly related to Hamming weight:

$$\mathscr{C}(t) = \epsilon\, \mathsf{H}(w) + \ell \,, \tag{4}$$

for some hardware-dependent constants ϵ and ℓ, and where w is the n-bit word manipulated at time period t.

Define, for $n \geq 1$,[†]

$$E_n = 2^{-2n} \sum_{w,r \in \{0,1\}^n} |\mathsf{H}(w \oplus r) - \mathsf{H}(r)| \ .$$

[†] Note that the expression of E_n simplifies to $E_n = 2^{-2n} \sum_{w,r \in \{0,1\}^n} |\mathsf{H}(w) - \mathsf{H}(r)|$.

Then, with the notations of Section 2.3, we get

$$\begin{aligned}\left\langle|\mathscr{C}(\tau_2)-\mathscr{C}(\tau_1)|\right\rangle_{\substack{w,r\in\{0,1\}^n\\ g(w)=0}} &= |\epsilon|\left\langle|\mathsf{H}(w\oplus r)-\mathsf{H}(r)|\right\rangle_{\substack{w,r\in\{0,1\}^n\\ g(w)=0}}\\ &= |\epsilon|\left\langle|\mathsf{H}(w\oplus r)-\mathsf{H}(r)|\right\rangle_{w,r\in\{0,1\}^{n-1}}\\ &= |\epsilon|\,E_{n-1}\ . \end{aligned} \tag{5}$$

Moreover, since

$$\begin{aligned}E_n &= \frac{1}{2}\left\langle|\mathsf{H}(w\oplus r)-\mathsf{H}(r)|\right\rangle_{\substack{w,r\in\{0,1\}^n\\ g(w)=0}} + \frac{1}{2}\left\langle|\mathsf{H}(w\oplus r)-\mathsf{H}(r)|\right\rangle_{\substack{w,r\in\{0,1\}^n\\ g(w)=1}}\\ &= \frac{1}{2}\,E_{n-1} + \frac{1}{2}\left\langle|\mathsf{H}(w\oplus r)-\mathsf{H}(r)|\right\rangle_{\substack{w,r\in\{0,1\}^n\\ g(w)=1}},\end{aligned}$$

we also get

$$\left\langle|\mathscr{C}(\tau_2)-\mathscr{C}(\tau_1)|\right\rangle_{\substack{w,r\in\{0,1\}^n\\ g(w)=1}} = |\epsilon|\,(2E_n - E_{n-1})\ . \tag{6}$$

Subtracting Eqs (6) and (5), we obtain

$$\begin{aligned}\overline{\mathfrak{D}_2} &:= \left\langle|\mathscr{C}(\tau_2)-\mathscr{C}(\tau_1)|\right\rangle_{\substack{w,r\in\{0,1\}^n\\ g(w)=1}} - \left\langle|\mathscr{C}(\tau_2)-\mathscr{C}(\tau_1)|\right\rangle_{\substack{w,r\in\{0,1\}^n\\ g(w)=0}}\\ &= 2|\epsilon|\,(E_n - E_{n-1})\ .\end{aligned} \tag{7}$$

It turns out that there is a nice closed formula for E_n:

Proposition 1. *For any integer $n \geq 1$, we have*

$$E_n = 2^{-2n}\,n\binom{2n}{n}\ . \tag{8}$$

Proof. From $\sum_{t=0}^{2n}\binom{2n}{t} = 2^{2n}$ and since $\sum_{t=0}^{n}\binom{2n}{t} = \sum_{t=n}^{2n}\binom{2n}{t}$, we get

$$\sum_{t=0}^{n}\tbinom{2n}{t} = \tfrac{1}{2}\big(2^{2n} + \tbinom{2n}{n}\big) = 2^{2n-1} + \tbinom{2n-1}{n}\,,$$

and similarly, $\sum_{t=0}^{n-1}\binom{2n-1}{t} = \frac{1}{2}\sum_{t=0}^{2n-1}\binom{2n-1}{t} = 2^{2n-2}$. Hence, by Lemma 1 (on page 6), we have

$$\begin{aligned}2^{2n}E_n &= \sum_{-n\leq t\leq n}|t|\tbinom{2n}{n-t} = 2\sum_{t=0}^{n}(n-t)\tbinom{2n}{t}\\ &= 2\sum_{t=0}^{n-1}n\tbinom{2n}{t} - 2\sum_{t=0}^{n-1}t\tbinom{2n}{t} = 2n\sum_{t=0}^{n-1}\tbinom{2n}{t} - 4n\sum_{t=0}^{n-2}\tbinom{2n-1}{t}\\ &= 2n\Big(\big[2^{2n-1} + \tbinom{2n-1}{n} - \tbinom{2n}{n}\big] - 2\big[2^{2n-2} - \tbinom{2n-1}{n-1}\big]\Big)\\ &= 2n\tbinom{2n-1}{n} = n\tbinom{2n}{n}\ .\end{aligned}$$

□

Therefore $E_n - E_{n-1} = 2^{-2n+1}\binom{2n-2}{n-1}$, and we find the *exact* value of $\overline{\mathfrak{D}_2}$ and its asymptotic behavior, using equation (7):

$$\overline{\mathfrak{D}_2} = |\epsilon|\, 2^{-2n+2}\binom{2n-2}{n-1} \approx \frac{|\epsilon|}{\sqrt{\pi n}} \tag{9}$$

as asymptotic value for $\overline{\mathfrak{D}_2}$.[‡] If there are sufficiently many acquisitions, it also represents an asymptotic value for $\overline{\Delta_2}(\hat{s})$ (cf. Eq. (3)). This approximation for $\overline{\mathfrak{D}_2}$ is already close for small values of n. It follows, for example, that one may expect peaks of size $\approx \frac{|\epsilon|}{\sqrt{\pi 16}} = 0.141\,|\epsilon|$, for $n = 16$ (this has to be compared with the exact value of $0.144\,|\epsilon|$, see Table 1).

3.2 Optimizing peak values

The higher-order DPA attacks, as described in Section 2.3, use the absolute difference between the power consumption at different time periods as the basic quantity for the analysis. This quantity fits well with the Hamming-weight model and we have shown how to determine the expected peak value in an exact way. A natural question is which other quantities can be used, and in particular, which quantities give rise to higher peak values.

In this section we analyze the peak values obtained for the following generalization of Eq. (7):

$$\overline{\mathfrak{D}_2}^{(\alpha)} = \big\langle |\mathscr{C}(\tau_2) - \mathscr{C}(\tau_1)|^\alpha \big\rangle_{\substack{w,r\in\{0,1\}^n \\ g(w)=1}} - \big\langle |\mathscr{C}(\tau_2) - \mathscr{C}(\tau_1)|^\alpha \big\rangle_{\substack{w,r\in\{0,1\}^n \\ g(w)=0}}$$

for arbitrary α. Extending our basic result for this case yields

$$\overline{\mathfrak{D}_2}^{(\alpha)} = 2|\epsilon|^\alpha \left(E_n^{(\alpha)} - E_{n-1}^{(\alpha)}\right), \tag{10}$$

where $E_n^{(\alpha)}$ is defined as:

$$E_n^{(\alpha)} = 2^{-2n} \sum_{w,r\in\{0,1\}^n} |\mathsf{H}(w \oplus r) - \mathsf{H}(r)|^\alpha \ . \tag{11}$$

Next we show how to find closed formulas for $E_n^{(\alpha)}$ for various values of α, from which we may then determine the corresponding expected peak values.

Lemma 1. *For any integer $n \geq 1$,*

$$\sum_{w,r\in\{0,1\}^n} |\mathsf{H}(w \oplus r) - \mathsf{H}(r)|^\alpha = \sum_{-n\leq t\leq n} |t|^\alpha \binom{2n}{n-t} \ .$$

[‡] This expression corrects the analysis given in [12] and in the subsequent papers.

Proof.

$$\sum_{w,r\in\{0,1\}^n} |\mathsf{H}(w\oplus r)-\mathsf{H}(r)|^\alpha = \sum_h \sum_{\substack{r\in\{0,1\}^n\\ \mathsf{H}(r)=h}} \sum_i \sum_{\substack{w\in\{0,1\}^n\\ \mathsf{H}(w\oplus r)=i}} |\mathsf{H}(w\oplus r)-\mathsf{H}(r)|^\alpha$$
$$= \sum_{h=0}^{n} \tbinom{n}{h} \sum_{i=0}^{n} \tbinom{n}{i} |i-h|^\alpha = \sum_{-n\le t\le n} |t|^\alpha \sum_{i=0}^{n-t} \tbinom{n}{t+i}\tbinom{n}{i}$$
$$= \sum_{-n\le t\le n} |t|^\alpha \tbinom{2n}{n-t} \ .$$

□

Theorem 1. *For any integer n and $\alpha \ge 0$:*

$$E_n^{(\alpha)} = \begin{cases} 1\,, & \alpha = 0\,, \\ 2^{-2n}\, n\tbinom{2n}{n}\,, & \alpha = 1\,, \\ n\left(nE_n^{(\alpha-2)} - \left(n-\frac{1}{2}\right)E_{n-1}^{(\alpha-2)}\right)\,, & \alpha \ge 2\,. \end{cases}$$

Proof. By induction on α. Cases $\alpha = 0$ and $\alpha = 1$ have been proved already. For the case $\alpha \ge 2$, we have:

$$\begin{aligned} E_n^{(\alpha)} &= 2^{-2n} \sum_{-n\le t\le n} |t|^\alpha \tbinom{2n}{n-t} = 2^{-2n} \sum_{0\le t\le 2n} |n-t|^\alpha \tbinom{2n}{t} \\ &= 2^{-2n} \sum_{0\le t\le 2n} |n-t|^{\alpha-2}\,(n-t)^2 \tbinom{2n}{t} \\ &= n^2\, 2^{-2n} \sum_{0\le t\le 2n} |n-t|^{\alpha-2} \tbinom{2n}{t} - 2^{-2n} \sum_{0\le t\le 2n} |n-t|^{\alpha-2}(2n-t)t\tbinom{2n}{t} \\ &= n^2 E_n^{(\alpha-2)} - 2^{-2n} \sum_{1\le t\le 2n-1} |n-t|^{\alpha-2}\, 2n(2n-1)\tbinom{2n-2}{t-1} \\ &= n\Big(nE_n^{(\alpha-2)} - 2(2n-1)2^{-2n} \sum_{0\le u\le 2(n-1)} |n-1-u|^{\alpha-2}\tbinom{2(n-1)}{u}\Big) \\ &= n\big(nE_n^{(\alpha-2)} - (n-\tfrac{1}{2})E_{n-1}^{(\alpha-2)}\big) \ . \end{aligned}$$

□

Alternatively, we have the following equivalent formulation:

Theorem 2. *For any integer n and $\beta \ge 0$:*

$$E_n^{(2\beta)} = \mathscr{P}_\beta(n) \quad \textit{and} \quad E_n^{(2\beta+1)} = \mathscr{Q}_\beta(n)\, 2^{-2n} n \binom{2n}{n}$$

where

$$\mathscr{P}_\beta(n) = \begin{cases} 1\,, & \beta = 0\,, \\ n\big(n\mathscr{P}_{\beta-1}(n) - (n-\frac{1}{2})\mathscr{P}_{\beta-1}(n-1)\big)\,, & \beta \ge 1\,, \end{cases} \tag{12}$$

and

$$\mathscr{Q}_\beta(n) = \begin{cases} 1\,, & \beta = 0\,, \\ n\big(n\mathscr{Q}_{\beta-1}(n) - (n-1)\mathscr{Q}_{\beta-1}(n-1)\big)\,, & \beta \ge 1\,. \end{cases} \tag{13}$$

□

Polynomials $\mathscr{Q}_\beta(n)$ (resp. $\mathscr{P}_\beta(n)$) are known as the Gandhi polynomials (resp. 'companion' Gandhi polynomials) — except that the Gandhi polynomials do not have alternating signs for the coefficients (but this difference is not essential). See [1, 2] and the references therein.

For illustration, we list below the so-obtained expression for $E_n^{(\alpha)}$ for the first few values of α.

Proposition 2. *We have:*

$$\begin{aligned}
E_n^{(0)} &= \mathscr{P}_0(n) && = 1 \\
E_n^{(1)} &= \mathscr{Q}_0(n)\, 2^{-2n}\, n\tbinom{2n}{n} && = 2^{-2n}\, n\tbinom{2n}{n} \\
E_n^{(2)} &= \mathscr{P}_1(n) && = n/2 \\
E_n^{(3)} &= \mathscr{Q}_1(n)\, 2^{-2n}\, n\tbinom{2n}{n} && = 2^{-2n}\, n^2\tbinom{2n}{n} \\
E_n^{(4)} &= \mathscr{P}_2(n) && = n(3n-1)/4 \\
E_n^{(5)} &= \mathscr{Q}_2(n)\, 2^{-2n}\, n\tbinom{2n}{n} && = 2^{-2n}\, n^2(2n-1)\tbinom{2n}{n} \\
E_n^{(6)} &= \mathscr{P}_3(n) && = n(15n^2-15n+4)/8 \\
E_n^{(7)} &= \mathscr{Q}_3(n)\, 2^{-2n}\, n\tbinom{2n}{n} && = 2^{-2n}\, n^2(6n^2-8n+3)\tbinom{2n}{n} \\
E_n^{(8)} &= \mathscr{P}_4(n) && = n(105n^3-210n^2+147n-34)/16 \\
E_n^{(9)} &= \mathscr{Q}_4(n)\, 2^{-2n}\, n\tbinom{2n}{n} && = 2^{-2n}\, n^2(24n^3-60n^2+54n-17)\tbinom{2n}{n} \\
E_n^{(10)} &= \mathscr{P}_5(n) && = n(945n^4-3150n^3+4095n^2-2370n+496)/32\,.
\end{aligned}$$

□

As a result, we find the following peak values:

$$\begin{aligned}
\overline{\mathfrak{D}_2}^{(\alpha)} &\approx |\epsilon|\, \tfrac{1}{\sqrt{\pi}}\, n^{-1/2}, && \text{for } \alpha = 1 \\
\overline{\mathfrak{D}_2}^{(\alpha)} &= |\epsilon|^2, && \text{for } \alpha = 2 \\
\overline{\mathfrak{D}_2}^{(\alpha)} &\approx |\epsilon|^3\, \tfrac{1}{\sqrt{\pi}}\, n^{1/2}, && \text{for } \alpha = 3 \\
\overline{\mathfrak{D}_2}^{(\alpha)} &\approx |\epsilon|^4\, \tfrac{3}{\sqrt{\pi}}\, n\,, && \text{for } \alpha = 4\,.
\end{aligned}$$

So the pattern that emerges is:

$$\overline{\mathfrak{D}_2}^{(\alpha)} \approx |\epsilon|^\alpha\, c_\alpha\, n^{(\alpha-2)/2}\,, \tag{14}$$

where c_α denotes a constant depending on α. Thus depending on the values of $|\epsilon|$ and n it may determined for which α the largest peak value is reached.

In Table 1, we tabulate the expected height, $\overline{\mathfrak{D}_2}^{(\alpha)}$, of the second-order DPA peaks for various values of n and α. We see for $\alpha = 1$ that larger values for n

(i.e., the bit-length manipulated by the processor) yields lower (second-order) DPA peaks. More surprisingly, for $\alpha = 2$, the height of the DPA peaks does not depend on n, and for larger values of α (two last columns in Table 1) the height increases with the value of n.

Table 1. Exact values of $\overline{\mathfrak{D}_2}^{(\alpha)}$ for some common sizes of n.

n	$\alpha = 1$	$\alpha = 2$	$\alpha = 3$	$\alpha = 4$
8	$0.209\,\lvert\epsilon\rvert$	$\lvert\epsilon\rvert^2$	$4.61\,\lvert\epsilon\rvert^3$	$22\,\lvert\epsilon\rvert^4$
16	$0.144\,\lvert\epsilon\rvert$	$\lvert\epsilon\rvert^2$	$6.65\,\lvert\epsilon\rvert^3$	$46\,\lvert\epsilon\rvert^4$
32	$0.101\,\lvert\epsilon\rvert$	$\lvert\epsilon\rvert^2$	$9.49\,\lvert\epsilon\rvert^3$	$94\,\lvert\epsilon\rvert^4$
64	$0.071\,\lvert\epsilon\rvert$	$\lvert\epsilon\rvert^2$	$13.48\,\lvert\epsilon\rvert^3$	$190\,\lvert\epsilon\rvert^4$
160	$0.045\,\lvert\epsilon\rvert$	$\lvert\epsilon\rvert^2$	$21.37\,\lvert\epsilon\rvert^3$	$478\,\lvert\epsilon\rvert^4$
256	$0.035\,\lvert\epsilon\rvert$	$\lvert\epsilon\rvert^2$	$27.05\,\lvert\epsilon\rvert^3$	$766\,\lvert\epsilon\rvert^4$
512	$0.025\,\lvert\epsilon\rvert$	$\lvert\epsilon\rvert^2$	$38.28\,\lvert\epsilon\rvert^3$	$1534\,\lvert\epsilon\rvert^4$
1024	$0.018\,\lvert\epsilon\rvert$	$\lvert\epsilon\rvert^2$	$54.15\,\lvert\epsilon\rvert^3$	$3070\,\lvert\epsilon\rvert^4$

The results listed in Table 1 are useful for practical purposes assuming that sufficiently many acquisitions are available. To see how fast $\overline{\mathfrak{D}_2}^{(\alpha)}$ actually converges to its expected value as a function of α, we determine the relevant signal-to-noise ratio (SNR), following, e.g., [13]. In the present paper, we are concerned with so-called algorithmic noise only, so the value of SNR tells us how many traces we need (for different values of input data x) to get a successful DPA attack.

Consider the following random variable D:

$$D := D(\hat{s}) = (2g(\hat{w}) - 1)\,|\epsilon|^\alpha\,|\mathsf{H}(w \oplus r) - \mathsf{H}(r)|^\alpha\,,$$

where r is a uniformly random n-bit string, $w = \mathtt{I}(x, s)$ with x representing the input data, and $\hat{w}$ is the outcome corresponding to the guessed secret $\hat{s}$, that is, $\hat{w} = \mathtt{I}(x, \hat{s})$. The process of evaluating quantity $\overline{\Delta_2}(\hat{s})$ (and its generalization for arbitrary α) may be viewed as sampling the random variable D.

The expected value of D is equal to $\overline{\mathfrak{D}_2}^{(\alpha)}$, and since $D^2 = |\epsilon|^{2\alpha}\,|H(w \oplus r) - H(r)|^{2\alpha}$, we obtain for the variance of D:

$$\mathrm{var}(D) = \langle D^2 \rangle - \langle D \rangle^2 = |\epsilon|^{2\alpha}\big(E_n^{(2\alpha)} - 4(E_n^{(\alpha)} - E_{n-1}^{(\alpha)})^2\big)\ .$$

Noting that the variance of D is independent of $\hat{s}$, we take as the relevant signal-to-noise ratio (for one signal):

$$\mathrm{SNR} := \frac{\langle D \rangle}{\sqrt{\mathrm{var}(D)}} = \frac{2\big(E_n^{(\alpha)} - E_{n-1}^{(\alpha)}\big)}{\sqrt{E_n^{(2\alpha)} - 4\big(E_n^{(\alpha)} - E_{n-1}^{(\alpha)}\big)^2}}\ . \tag{15}$$

From this formula for SNR, we get the following results. Firstly, SNR is independent of ϵ; this corresponds to the results found in, e.g., [13], where SNR

is also independent of ϵ if there is *no* (non-algorithmic) noise. Secondly, SNR drops off to 0 quickly as n gets larger, which was also observed in [13]. Finally, however, by evaluating SNR for fixed values of n, $n \geq 3$, searching for the optimal value of α we have found that SNR is maximized consistently at $\alpha = 3$, where SNR at $\alpha = 3$ is about 1.55 times higher than at $\alpha = 1$.

3.3 Analysis of other correlated operations

So far, we have focused on the use of $\oplus$-masking, which is the basic way of implementing data whitening. Since many attacks are proposed to defeat $\oplus$-masking it is conceivable that implementors try other forms of masking, hoping to avoid such DPA attacks. For example, $w \oplus r$ may be computed using the following formula: $\overline{(w \wedge r) \vee (\bar{w} \wedge \bar{r})}$. Our calculations show that in such a case the medicine is worse than the disease, as a second-order DPA for the $\wedge$ operation yields better peaks than for $\oplus$ operation.

We illustrate this by determining $\overline{\mathfrak{D}_2}$ (cf. Eq. (7)) where τ_1 is the time period when r is manipulated and τ_2 is the time period when $w \wedge r$ is manipulated.

We have:

$$\begin{aligned}
&\left\langle |\mathscr{C}(\tau_2) - \mathscr{C}(\tau_1)| \right\rangle_{\substack{w,r\in\{0,1\}^n \\ g(w)=0}} = |\epsilon| \left\langle |\mathsf{H}(w \wedge r) - \mathsf{H}(r)| \right\rangle_{\substack{w,r\in\{0,1\}^n \\ g(w)=0}} \\
&= \frac{|\epsilon|}{2} \Big(\left\langle |\mathsf{H}(w \wedge r) - \mathsf{H}(r)| \right\rangle_{\substack{w,r\in\{0,1\}^n \\ g(w)=0, g(r)=0}} + \left\langle |\mathsf{H}(w \wedge r) - \mathsf{H}(r)| \right\rangle_{\substack{w,r\in\{0,1\}^n \\ g(w)=0, g(r)=1}} \Big) \\
&= \frac{|\epsilon|}{2} \Big(\left\langle |\mathsf{H}(w \wedge r) - \mathsf{H}(r)| \right\rangle_{w,r\in\{0,1\}^{n-1}} + \left\langle |\mathsf{H}(w \wedge r) - \mathsf{H}(r) - 1| \right\rangle_{w,r\in\{0,1\}^{n-1}} \Big) \ .
\end{aligned}$$

and

$$\begin{aligned}
\left\langle |\mathscr{C}(\tau_1) - \mathscr{C}(\tau_2)| \right\rangle_{\substack{w,r\in\{0,1\}^n \\ g(w)=1}} &= |\epsilon| \left\langle |\mathsf{H}(w \wedge r) - \mathsf{H}(r)| \right\rangle_{\substack{w,r\in\{0,1\}^n \\ g(w)=1}} \\
&= |\epsilon| \left\langle |\mathsf{H}(w \wedge r) - \mathsf{H}(r)| \right\rangle_{w,r\in\{0,1\}^{n-1}} \ .
\end{aligned}$$

Therefore, we get

$$\overline{\mathfrak{D}_2} = \frac{|\epsilon|}{2} \Big(\left\langle |\mathsf{H}(w \wedge r) - \mathsf{H}(r)| \right\rangle_{w,r\in\{0,1\}^{n-1}} - \left\langle |\mathsf{H}(w \wedge r) - \mathsf{H}(r) - 1| \right\rangle_{w,r\in\{0,1\}^{n-1}} \Big) \ .$$

To find the peak values we use the following lemma.

Lemma 2. *For any integer $n \geq 1$,*

$$2^{-2n} \sum_{w,r\in\{0,1\}^n} |\mathsf{H}(w \wedge r) - \mathsf{H}(r)| = \frac{n}{4} \, ,$$

and

$$2^{-2n} \sum_{w,r\in\{0,1\}^n} |\mathsf{H}(w \wedge r) - \mathsf{H}(r) - 1| = \frac{n}{4} + 1 \ .$$

Proof. We only prove the first part.

$$\begin{aligned}
\sum_{w,r\in\{0,1\}^n} |\mathsf{H}(w\wedge r)-\mathsf{H}(r)| &= \sum_h \sum_{\substack{r\in\{0,1\}^n\\ \mathsf{H}(r)=h}} \sum_i \sum_{\substack{w\in\{0,1\}^n\\ \mathsf{H}(w\wedge r)=i}} |\mathsf{H}(w\wedge r)-\mathsf{H}(r)| \\
&= \sum_{h=0}^{n} \tbinom{n}{h} 2^{n-h} \sum_{i=0}^{h} \tbinom{h}{i} |i-h| = \sum_{h=0}^{n} \tbinom{n}{h} 2^{n-h} \sum_{i=0}^{h} \tbinom{h}{i} (h-i) \\
&= \sum_{h=0}^{n} \tbinom{n}{h} 2^{n-h}\, h \sum_{i=0}^{h} \left[\tbinom{h}{i} - \tbinom{h-1}{i-1}\right] = \sum_{h=0}^{n} \tbinom{n}{h} 2^{n-h}\, h \sum_{i=0}^{h} \tbinom{h-1}{i} \\
&= \sum_{h=0}^{n} \tbinom{n}{h} 2^{n-h} h\, 2^{h-1} = 2^{n-1} \sum_{h=0}^{n} \tbinom{n}{h} h = n\, 2^{2n-2} \ .
\end{aligned}$$

□

Hence, the resulting expected peak value becomes

$$\overline{\mathfrak{D}_2} = -|\epsilon|/2 \ . \tag{16}$$

This results in a higher peak value compared to the peak for $\oplus$-masking (cf. Eq. (9)).

Other binary operations such as logical or can be handled as well using basic properties such as $\mathsf{H}(w \vee r) = \mathsf{H}(w) + \mathsf{H}(r) - \mathsf{H}(w \wedge r)$.

4 Extension to the Hamming Distance Model

The basic assumption for the Hamming-weight model is that the power consumption for an operation on some data word w is linearly related to $\mathsf{H}(w)$. In terms of electronics this would mean, however, that the hardware state just prior to the moment that a data word is handled is (re)set to all-zero (or, all-one). This is the case only for a few types of electronic circuits, e.g., those that use precharged logic.

In many cases it is necessary to assume that the actual power consumption is linearly related to the Hamming *distance* to an (unknown) hardware state that existed just prior the moment when data word w is handled. For instance, each time an instruction `I` within a fixed program is executed involving w, what actually happens is that the CPU must fetch the opcode from memory (by sending the program counter over the bus and receiving the opcode in return, and decoding it). The values of the program counter and the opcode are fixed (for `I`) and therefore the power consumption incurred by transferring these values over the bus is the same each time `I` is executed. Subsequently, when w is handled, the energy required for transferring w over the bus will be related to the number of bits that need to *switched*. See [5] for more details.

So, we assume that there exist two reference values R_1 and R_2, that represent the states prior to the time periods τ_1 and τ_2, respectively. The power consumption at these time periods is therefore related to $\mathsf{H}(R_1 \oplus r)$ and $\mathsf{H}(R_2 \oplus w \oplus r)$, respectively.

We now show that the same DPA trace as before will enable us to recover the secret value, without using *any* knowledge about the values of R_1 and R_2.

Let R_1' denote R_1 but with the bit indicated by g omitted, and similarly for R_2. Let also $\delta = g(R_2) - g(R_1)$. Then, noting that for any fixed R_1 and R_2, and for any δ:

$$\begin{aligned}\left\langle \left| \mathsf{H}(R_2 \oplus w \oplus r) - \mathsf{H}(R_1 \oplus r) + \delta \right| \right\rangle &= \left\langle \left| \mathsf{H}(w \oplus r) - \mathsf{H}(r) + \delta \right| \right. \\ &= \left\langle \left| \mathsf{H}(w) - \mathsf{H}(r) + \delta \right| \right. ,\end{aligned}$$

it follows that

$$\begin{aligned}\left\langle \left| \mathscr{C}(\tau_2) - \mathscr{C}(\tau_1) \right| \right\rangle_{\substack{w,r\in\{0,1\}^n \\ g(w)=0}} &= |\epsilon| \left\langle \left| \mathsf{H}(R_2 \oplus w \oplus r) - \mathsf{H}(R_1 \oplus r) \right| \right\rangle_{\substack{w,r\in\{0,1\}^n \\ g(w)=0}} \\ &= \frac{|\epsilon|}{2} \Big(\left\langle \left| \mathsf{H}(R_2 \oplus w \oplus r) - \mathsf{H}(R_1 \oplus r) \right| \right\rangle_{\substack{w,r\in\{0,1\}^n \\ g(w)=0, g(r)=0}} + \\ &\qquad \left\langle \left| \mathsf{H}(R_2 \oplus w \oplus r) - \mathsf{H}(R_1 \oplus r) \right| \right\rangle_{\substack{w,r\in\{0,1\}^n \\ g(w)=0, g(r)=1}} \Big) \\ &= \frac{|\epsilon|}{2} \Big(\left\langle \left| \mathsf{H}(R_2' \oplus w \oplus r) - \mathsf{H}(R_1' \oplus r) + \delta \right| \right\rangle_{w,r\in\{0,1\}^{n-1}} + \\ &\qquad \left\langle \left| \mathsf{H}(R_2' \oplus w \oplus r) - \mathsf{H}(R_1' \oplus r) - \delta \right| \right\rangle_{w,r\in\{0,1\}^{n-1}} \Big) \\ &= |\epsilon| \left\langle \left| \left| \mathsf{H}(w) - \mathsf{H}(r) + |\delta| \right| \right\rangle_{w,r\in\{0,1\}^{n-1}} .\end{aligned}$$

Similarly, letting $\delta' = 1 - g(R_2) - g(R_1)$, we have

$$\begin{aligned}\left\langle \left| \mathscr{C}(\tau_2) - \mathscr{C}(\tau_1) \right| \right\rangle_{\substack{w,r\in\{0,1\}^n \\ g(w)=1}} &= |\epsilon| \left\langle \left| \mathsf{H}(R_2 \oplus w \oplus r) - \mathsf{H}(R_1 \oplus r) \right| \right\rangle_{\substack{w,r\in\{0,1\}^n \\ g(w)=1}} \\ &= \frac{|\epsilon|}{2} \Big(\left\langle \left| \mathsf{H}(R_2 \oplus w \oplus r) - \mathsf{H}(R_1 \oplus r) \right| \right\rangle_{\substack{w,r\in\{0,1\}^n \\ g(w)=1, g(r)=0}} + \\ &\qquad \left\langle \left| \mathsf{H}(R_2 \oplus w \oplus r) - \mathsf{H}(R_1 \oplus r) \right| \right\rangle_{\substack{w,r\in\{0,1\}^n \\ g(w)=1, g(r)=1}} \Big) \\ &= \frac{|\epsilon|}{2} \Big(\left\langle \left| \mathsf{H}(R_2' \oplus w \oplus r) - \mathsf{H}(R_1' \oplus r) + \delta' \right| \right\rangle_{w,r\in\{0,1\}^{n-1}} + \\ &\qquad \left\langle \left| \mathsf{H}(R_2' \oplus w \oplus r) - \mathsf{H}(R_1' \oplus r) - \delta' \right| \right\rangle_{w,r\in\{0,1\}^{n-1}} \Big) \\ &= |\epsilon| \left\langle \left| \left| \mathsf{H}(w) - \mathsf{H}(r) + |\delta'| \right| \right\rangle_{w,r\in\{0,1\}^{n-1}} .\end{aligned}$$

Define, for $n \geq 1$,

$$E_{n,m} = 2^{-2n} \sum_{w,r\in\{0,1\}^n} \left| \mathsf{H}(w) - \mathsf{H}(r) + m \right| . \tag{17}$$

These sums may be evaluated using the following recurrence relation:

$$\begin{aligned} f(0,m) &= |m| \\ f(n,m) &= f(n-1,m-1) + 2f(n-1,m) + f(n-1,m+1) \end{aligned}$$

where

$$f(n,m) = \sum_{w,r \in \{0,1\}^n} \left| \mathsf{H}(w) - \mathsf{H}(r) + m \right|,$$

for any integers n, m, $n \geq 0$.

In particular, the first few terms are:

$$f(n,0) = n\binom{2n}{n}, \quad f(n,1) = (n+1)\binom{2n}{n}, \quad f(n,2) = \frac{n^2+5n+2}{n+1}\binom{2n}{n} .$$

Noting that $|\delta'| = 1 - |\delta|$, we then have:

$$\overline{\mathfrak{D}_2} = |\epsilon|(E_{n-1,1-|\delta|} - E_{n-1,|\delta|}), \tag{18}$$

which generalizes the previous results (compare with Eq. (7)).

If $g(R_1) = g(R_2)$ we get the same result as before. If $g(R_1) = 1 - g(R_2)$ we get the same result as before, except that the sign is inverted. So we are actually able to get 1 bit of information on the values R_1 and R_2 as well.

By varying the selection function g to target other bits, all the bits of $R_1 \oplus R_2$ can be recovered this way.

5 Experimental Results and Observations

The experimental validation of our results led us to choose the RC6-$w/r/b$ block-cipher [15], in which the word size $w = n$, the number of rounds r and the length b of the encryption key in bytes are easily customizable. Besides, RC6 contains no fixed-length substitution boxes that would have limited the testing of the range $n \in [8, 1024]$. Arithmetic operations $(+, -, \times)$ are defined modulo 2^n. The encryption software takes as input a plaintext $m \in \{0,1\}^{4n}$ stored in four n-bit input registers A, B, C, D, and an extended key $S_0, \ldots, S_{2r+3}$ where $S_i \in \{0,1\}^n$.

---

1. $B = B + S_0$ and $D = D + S_1$
2. for $i = 1$ to r do
   (a) $t = (B \times (2B+1)) \lll \log_2 n$ and $u = (D \times (2D+1)) \lll \log_2 n$
   (b) $A = ((A \oplus t) \lll u) + S_{2i}$ and $C = ((C \oplus u) \lll t) + S_{2i+1}$
   (c) $(A, B, C, D) = (B, C, D, A)$
3. $A = A + S_{2r+2}$ and $C = C + S_{2r+3}$

Fig. 1. Encryption with RC6-$n/r/b$.

The protection against first-order attacks relies on a data whitening technique mixing boolean and two forms of arithmetic masking. We denote by $\langle\!\langle x \rangle\!\rangle$ the variable x randomized by a boolean mask, i.e., $\langle\!\langle x \rangle\!\rangle = (x \oplus \rho, \rho)$ for some $\rho \in \{0,1\}^n$. Similarly, we note $[\![x]\!] = (x + \rho', \rho')$ an arithmetic randomization

of x modulo 2^n. We make use of Goubin's conversion technique [6] to compute $[\![x]\!]$ from $\langle\!\langle x \rangle\!\rangle$ securely, and on a technique due to Coron and Tchoulkine to come back to boolean masking $[\![x]\!] \mapsto \langle\!\langle x \rangle\!\rangle$ (provably) without first-order information leakage. Conversions $[\![x]\!] \mapsto \langle\!\langle x \rangle\!\rangle$ rely on a precomputed look-up table of constants embedded into the code. We chose to re-randomize the output after each conversion.

Dealing with rotations. RC6 involves circular rotations with variable offset denoted by $x \lll u$ such that $2^n - 1 \lll u = 2^n - 1$ for any $u \in \{0,1\}^n$ and $x \lll u = x \times 2^{u \bmod n} \bmod (2^n - 1)$ for any $x \neq 2^n - 1$ and any $u \in \{0,1\}^n$. We suggest a technique to compute $\langle\!\langle x \lll u \rangle\!\rangle$ from $\langle\!\langle x \rangle\!\rangle$ and $\langle\!\langle u \rangle\!\rangle$. First, we apply Goubin's conversion to $\langle\!\langle u \rangle\!\rangle$ with respect to word size $\log_2 n$, thereby computing securely $[\![u]\!]_o = (u - \alpha \bmod n, \alpha)$ where α is randomly taken from $\{0,1\}^{\log_2 n}$. Now if $\langle\!\langle x \rangle\!\rangle = (x \oplus \rho, \rho)$, we rotate the two shares twice each by $u - \alpha \bmod n$ and then α bit positions to the left:

$$\langle\!\langle x \rangle\!\rangle, [\![u]\!]_o \mapsto \langle\!\langle x \lll u \rangle\!\rangle = \langle\!\langle ((x \oplus \rho) \lll (u - \alpha)) \lll \alpha, (\rho \lll (u - \alpha)) \lll \alpha \rangle\!\rangle .$$

Description of our 1-st-order-protected RC6. We assume that the extended key is stored as $[\![S_0]\!], \ldots, [\![S_{2r+3}]\!]$ where each $S_i \in \{0,1\}^n$ is arithmetically randomized[§]. We show how to protect all the internal steps. The computation $B = B + S_0$ in Step 1 consists in randomly masking B into $[\![B]\!]$, computing $[\![B + S_0]\!]$ from $[\![S_0]\!], [\![B]\!]$ and then converting $[\![B]\!]$ into $\langle\!\langle B \rangle\!\rangle$. The same is done for D. We also randomly mask A and C into $\langle\!\langle A \rangle\!\rangle$ and $\langle\!\langle C \rangle\!\rangle$. In Step 2a, we convert $\langle\!\langle B \rangle\!\rangle \mapsto [\![B]\!]$, securely compute $[\![B \times (2B + 1)]\!]$ from $[\![B]\!] = (B_1, B_0)$ as $[\![B \times (2B + 1)]\!] = (B_1 \times (2B_1 + 1) - \gamma \times B_0, B_0 \times (4B_1 + 2B_0 + \gamma + 1))$ for a random $\gamma \in \{0,1\}^n$, convert this into $\langle\!\langle B \times (2B + 1) \rangle\!\rangle$ and rotate each share by $\log_2 n$ bits to get (t). We apply the same process to compute u. In Step 2b, we compute $\langle\!\langle A \rangle\!\rangle, \langle\!\langle t \rangle\!\rangle \mapsto \langle\!\langle A \oplus t \rangle\!\rangle$ and convert $\langle\!\langle u \rangle\!\rangle \mapsto [\![u]\!]_o$. Then from $\langle\!\langle A \oplus t \rangle\!\rangle, [\![u]\!]_o$ we get $\langle\!\langle (A \oplus t) \lll u \rangle\!\rangle$ which we convert into $[\![(A \oplus t) \lll u]\!]$ to add it to $[\![S_{2i}]\!]$. The result is stored as $[\![A]\!]$ which we convert into $\langle\!\langle A \rangle\!\rangle$. We do the same for register C. Lastly, Step 3 consists in converting $\langle\!\langle A \rangle\!\rangle$ into $[\![A]\!]$, adding it to $[\![S_{2r+2}]\!]$ and subtracting the two shares to get A (the same applies to C). From $\langle\!\langle B \rangle\!\rangle, \langle\!\langle D \rangle\!\rangle$, we recover B and D.

Attack strategy. Assuming that we know subkeys $S_0, \ldots, S_{2j-2}, S_{2j-1}$ for some $j \in [0, r-1]$, we recover subkeys S_{2j}, S_{2j+1} involved in the j-th round. As our technique applies to boolean masking, we focus on the last computation of Step 2b that converts $[\![A]\!] \mapsto \langle\!\langle A \rangle\!\rangle$ just after S_{2j} has been added to A. Noting $\langle\!\langle A \rangle\!\rangle = (A_1, A_0)$, we know that our conversion routine handles A_1 first and then A_0 two clock cycles later. This gives us an offset $\delta = 2$ and power traces will reveal S_{2j}. We apply the same technique to the conversion $[\![C]\!] \mapsto \langle\!\langle C \rangle\!\rangle$ in the same round to get S_{2j+1}. We mention for concreteness that similar strategies apply to Steps 1, 3 to get $S_0, S_1, S_{2r+2}, S_{2r+3}$.

[§] Each $[\![S_i]\!]$ involves its own randomness. In our implementation, $[\![S_i]\!]$ is re-randomized after use and updated in volatile memory for later calls to RC6.

Data Treatment and Observations. The execution of our code is simulated with `Mathematica 5.0` and power traces generated with $\epsilon = 1$ and $\ell = 0$. We chose $r = 8$ and $j = 5$ arbitrarily, $\alpha = 3$ to maximize the SNR, and the selection function $g(w)$ is the most significant bit of w. The value of $\langle\!\langle A \rangle\!\rangle$ when converting $[\![A]\!] \mapsto \langle\!\langle A \rangle\!\rangle$ in the 5-th round is identical to the value of $\langle\!\langle B \rangle\!\rangle$ before the conversion $\langle\!\langle B \rangle\!\rangle \mapsto [\![B]\!]$ in Step 2a of round 6. The conversion routine from boolean to arithmetic masking also admits an offset of two cycles. Therefore, power traces present two attack locations and two (potential) simultaneous peaks.

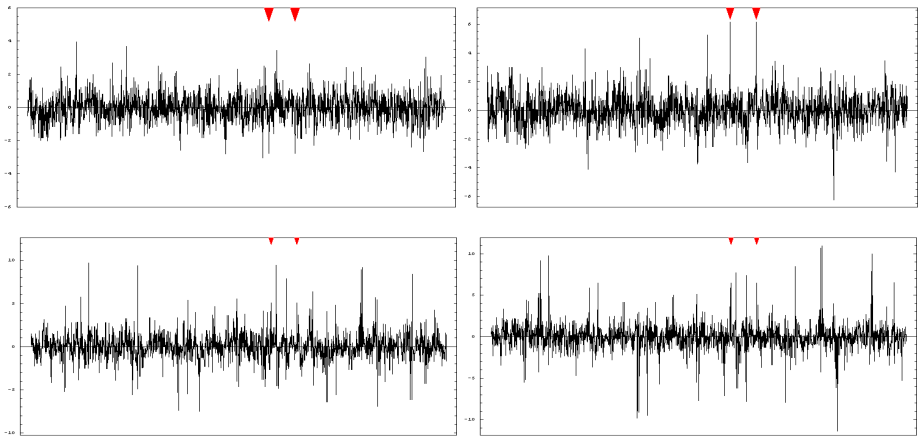

Fig. 2. Second-order DPA trace $\Delta_2(\hat{s}, t)$ with wrong guess $s \neq \hat{s}$ (left) and correct guess $s = \hat{s}$ (right). About 3000 power traces are enough to see peaks for $n = 8$ (top). More than 20000 power traces are necessary for $n = 16$ (bottom), indicating a loss of efficiency as n grows. Arrows indicate attack locations in the 5-th and 6-th rounds.

For $n = 8$, we measure peaks of height 5.52 to be compared with the theoretical value of 4.61. For $n = 16$, we observe peaks at 6.45 for an expected height of 6.65. Standard deviations are 0.82 and 1.60 respectively. This seems to confirm our theoretical results. Experiments for $n \geq 32$ show that a much larger number of power traces are needed to mount a practical attack.

6 Conclusions

We provided a formal exploration of second-order attacks by estimating the exact height of expected peaks. Our results allow to anticipate the efficiency of second-order attacks on a given hardware by providing a theoretical measurement of information leakage as a function of hardware-dependent parameters. We believe that our results constitute the theoretical basis of practical general-purpose second-order power attacks. We see many open issues for future research, in particular how to extend our results to take non-algorithmic noise into account.

References

1. ⟨http://www.research.att.com/projects/OEIS?Anum=A036970⟩. Triangle of coefficients of Gandhi polynomials. In *On-Line Encyclopedia of Integer Sequences*.
2. ⟨http://www.research.att.com/projects/OEIS?Anum=A083061⟩. Triangle of coefficients of a companion polynomial to the Gandhi polynomial. In *On-Line Encyclopedia of Integer Sequences*.
3. D. Agrawal, B. Archambeault, J. R. Rao, and P. Rohatgi. The EM Side-Channel(s). In *Cryptographic Hardware and Embedded Systems – CHES 2002*, vol. 2523 of *Lecture Notes in Computer Science*, pp. 29–45. Springer-Verlag, 2002.
4. G. Boros and V. Moll. *Irresistible Integrals: Symbolics, Analysis and Experiments in the Evaluation of Integrals*. Cambridge University Press, 2004.
5. É. Brier, C. Clavier, and F. Olivier. Correlation power analysis with a leakage model. In *Cryptographic Hardware and Embedded Systems – CHES 2004*, vol. 3156 of *Lecture Notes in Computer Science*, pp. 16–29. Springer-Verlag, 2004.
6. J.-S. Coron and L. Goubin. On Boolean and arithmetic masking against differential power analysis. In *Cryptographic Hardware and Embedded Systems – CHES 2000*, vol. 1965 of *Lecture Notes in Computer Science*, pp. 231–237. Springer-Verlag, 2000.
7. K. Gandolfi, C. Mourtel, and F. Olivier. Electromagnetic analysis: Concrete results. In *Cryptographic Hardware and Embedded Systems – CHES 2001*, vol. 2162 of *Lecture Notes in Computer Science*, pp. 251–261. Springer-Verlag, 2001.
8. P.C. Kocher, J. Jaffe, and B. Jun. Differential power analysis. In *Advances in Cryptology – CRYPTO '99*, vol. 1666 of *Lecture Notes in Computer Science*, pp. 388–397. Springer-Verlag, 1999.
9. M. Joye. Smart-card implementations of elliptic curve cryptography and DPA-type attacks. In *Smart Card Research and Advanced Applications VI*, pp. 115–125. Kluwer Academic Publishers, 2004.
10. D.E. Knuth. *The Art of Computer Programming (Vol. 1: Fundamental Algorithms)*. Addison Wesley, 3rd edition, 1997.
11. P.C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In *Advances in Cryptology – CRYPTO '96*, vol. 1109 of *Lecture Notes in Computer Science*, pp. 104–113. Springer-Verlag, 1996.
12. T.S. Messerges. Using second-order power analysis to attack DPA resistant software. In *Cryptographic Hardware and Embedded Systems – CHES 2000*, vol. 1965 of *Lecture Notes in Computer Science*, pp. 238–251. Springer-Verlag, 2000.
13. T.S. Messerges, E.A. Dabbish, and R.H. Sloan. Examining smart-card security under the threat of power analysis attacks. *IEEE Transactions on Computers*, **51**(5):541 552, 2002.
14. J.-J. Quisquater and D. Samyde. ElectroMagnetic Analysis (EMA): Measures and couter-measures for smard cards. In *Smart Card Programming and Security (E-smart 2001)*, vol. 2140 of *Lecture Notes in Computer Science*, pp. 200–210. Springer-Verlag, 2001.
15. R.L Rivest, M.J.B. Robshaw, R. Sideney, and Y.L. Yin. The RC6 block cipher. RSA Laboratories, v1.1, August 20, 1998.
16. J. Waddle and D. Wagner. Towards efficient second-order power analysis. In *Cryptographic Hardware and Embedded Systems – CHES 2004*, vol. 3156 of *Lecture Notes in Computer Science*, pp. 1–15. Springer-Verlag, 2004.