Hindawi Publishing Corporation
The Scientific World Journal
Volume 2015, Article ID 743618, 10 pages
http://dx.doi.org/10.1155/2015/743618

Research Article

Exploiting Small Leakages in Masks to Turn a Second-Order Attack into a First-Order Attack and Improved Rotating Substitution Box Masking with Linear Code Cosets

Alexander DeTrano,[1] Naghmeh Karimi,[2] Ramesh Karri,[1] Xiaofei Guo,[3] Claude Carlet,[4] and Sylvain Guilley[5,6]

[1]*New York University, New York, NY 10012, USA*
[2]*Rutgers University, New Brunswick, NJ 08901, USA*
[3]*Security Center of Excellence, Intel Corporation, Hillsboro, OR 97124, USA*
[4]*Paris 8 University, 93526 Saint-Denis, France*
[5]*Télécom ParisTech, 75634 Paris, France*
[6]*Secure-IC S.A.S., 35510 Cesson-Sévigné, France*

Correspondence should be addressed to Sylvain Guilley; sylvain.guilley@telecom-paristech.fr

Received 15 June 2015; Accepted 25 August 2015

Academic Editor: Makoto Nagata

Masking countermeasures, used to thwart side-channel attacks, have been shown to be vulnerable to mask-extraction attacks. State-of-the-art mask-extraction attacks on the Advanced Encryption Standard (AES) algorithm target S-Box recomputation schemes but have not been applied to scenarios where S-Boxes are precomputed offline. We propose an attack targeting precomputed S-Boxes stored in nonvolatile memory. Our attack targets AES implemented in software protected by a low entropy masking scheme and recovers the masks with 91% success rate. Recovering the secret key requires fewer power traces (in fact, by at least two orders of magnitude) compared to a classical second-order attack. Moreover, we show that this attack remains viable in a noisy environment or with a reduced number of leakage points. Eventually, we specify a method to enhance the countermeasure by selecting a suitable coset of the masks set.

1. Introduction

Traditionally, a cryptographic algorithm was considered secure if it withstood classical linear and differential cryptanalysis. A side-channel attack exploits physical characteristics of a device in order to recover secret information, such as the encryption key. Power dissipation and electromagnetic (EM) emanation side-channel attacks are of particular concern because of their low implementation cost, ease of use, and effectiveness in extracting secret information [1]. Power analysis attacks work because the amount of power (or EM emanations) dissipated by a device is dependent on the data being processed. The Advanced Encryption Standard (AES) is the standard symmetric key encryption specified by the National Institute of Standards and Technology (NIST) [2] and is also included in ISO/IEC 18033-3:2010 [3]. It is widely used in electronic systems such as automated teller machines, telecommunications, and virtual private networks. Traditional cryptanalysis cannot break AES. However, if AES is not carefully implemented, side-channel attacks can leak the secret key [1, 4–8].

1.1. Related Work. Masking variables is a well-known countermeasure [9–12] to protect against side-channel attacks. Sensitive variables are concealed by random variables. Masking comes in a variety of flavors; however, we consider only the *Boolean* type in this paper. Boolean masking splits a sensitive variable x into a number $(d+1)$ of shares by the exclusive-or (XOR) operation $x = x_0 \oplus \cdots \oplus x_d$. Each share is processed

independently so that the measured leakage depends on some random value, rather than the sensitive information. A first-order masking scheme uses one mask, whereas a dth-order masking scheme uses d masks. A $(d+1)$th-order attack targets the manipulation of $d+1$ manipulated variables that jointly depend on a secret value. A dth-order masking scheme can be broken by a $(d+1)$th-order attack [13]. Masking strategies can also be classified according to the amount of entropy used; intuitively, the more the entropy in the set of masks is, the more secure the implementations are against side-channel analysis. *Full Entropy Masking Schemes* (FEMS) draw masks from the entire mask set to conceal sensitive information [14]. In the case of AES, each plaintext byte is masked, and so each mask can take on all 256 values from $\mathbb{F}_2^8$. *Low Entropy Masking Schemes* (LEMS) instead draw masks from a reduced mask set, a strict subset of $\mathbb{F}_2^8$ [14, 15].

Masking the nonlinear portions of AES, that is, the substitution boxes (S-Boxes), can be costly. The masked S-Boxes can be calculated on the fly for each encryption [9], securely precomputed before encryption begins [16], or generated offline and stored in Read-Only Memory (ROM) or in Random Access Memory (RAM) [17]. The S-Box precomputation scheme suits AES, because the 16 S-Boxes are the same (unlike, e.g., the Data Encryption Standard—DES). However, the S-Box precomputation method significantly increases total encryption time. The masked S-Box is typically recalculated for every encryption and this S-Box recomputation can be as long as the entire AES operation, if not longer. For instance, the authors in [13] describe AES implementation that takes twice as long to encrypt a plaintext versus the equivalent unprotected version; 33% of the runtime is spent calculating the masked S-Box. The frequent reuse of the mask during the S-Box precomputation allows for horizontal attacks (deemed horizontal because multiple points along a single power trace are analyzed [18]), which exploit the high multiplicity of samples (namely, 256) to recover the mask [19–21].

Computing offline the entire set of masked S-Boxes (256 for FEMS) alleviates the extra runtime issue of S-Box recomputation but requires at least 64 kilobytes of memory which is beyond the capacity of embedded systems such as smartcards. LEMS offers a tradeoff between complexity and security. The space required for a LEMS using 16 masks out of 256 masks is that needed to store 16 S-Boxes (namely, 4 kilobytes of storage). Removing the need for lengthy masked S-Box precomputation, we notice that LEMS are less prone to attacks such as those described in [19–21]. Additional masks (as in high-order masking schemes) increase the complexity and area overhead of the design, since these extra masks have to be stored in memory or calculated at some point in time. Therefore, first-order masking schemes are the mainstream protection.

1.2. Contribution and Outline. Efficient first-order masking schemes (FEMS using S-Box precomputation or LEMS such as Rotating S-Box Masking [17]) reuse the same mask several times, typically at each S-Box call; therefore, a horizontal power analysis attack on 16 leakage points can reveal the mask. We show that the state-of-the-art mask-extraction attack [20] on S-Box precomputation can be retargeted towards masked AES implementation. Indeed, the attack presented in [19–21] is the core idea of this paper. At the time of writing, a similar attack was published on the DPA Contest website [22] by Nakai et al. We want to stress that both works were performed independently of each other. We therefore add value by exploring the attack parameters in order to gain a deeper understanding of the strength of the attack. This paper has three main contributions. First, we show that the attack can succeed even in the presence of noise: tiny information on the mask can be extracted, enabling a first-order attack in a second pass. Second, we find that this type of attack outperforms a classical second-order attack with respect to number of traces needed to recover the key. Third, we explore improvements of the code employed for masks of the Rotating S-Box Masking countermeasure to make the exploitation of the leakage more difficult.

The rest of the paper is organized as follows. Section 2 proposes the mask recovery attack and validates it using publicly available data. Section 3 discusses the attack results and attack parameters, compares the attack with a state-of-the-art second-order attack [23] in noisy environments, and proposes a countermeasure. Section 4 concludes the paper and opens some perspectives. The Appendix exhibits a constant Hamming weight code, but with resistance against only first-order attacks. The countermeasure presented in Section 3 and the tradeoff discussed in the Appendix are two noticeable contributions with respect to the preliminary conference version of this paper [24].

2. The Proposed Mask Recovery Attack

We describe the implemented countermeasure, power analysis, and the proposed attack.

2.1. Rotating S-Box Masking. A first-order masking countermeasure called *Rotating S-Box Masking* (RSM) [17] is shown in Figure 1. The dotted boxes represent the additional steps added to AES-256 by RSM. RSM is a Boolean-additive LEMS and uses a total of 16 public-knowledge masks, $m_{0-15} \in \mathcal{M} \subset \mathbb{F}_2^8$, one for each byte of plaintext. At the start of each encryption, a random offset $j \in [0 \cdots 15]$ is drawn. The offset can be thought of as the number of positions to cyclically left-rotate the base set of masks, $\mathcal{M}_0$. The set of masks with offset j is denoted by $\mathcal{M}_j$; for example, if the offset $j = 0$, then the masks are deployed in the following order: $\mathcal{M}_0 = m_0, m_1, m_2, \ldots, m_{14}, m_{15}$. Thus, only 16 possibilities exist for the order of the masks, since a shift greater than 15 simply wraps the set of masks around. The masks are then XORed with the plaintext, and this result is XORed with the first round key. The S-Box is replaced by 16 masked S-Boxes, where each S-Box corresponds to an offset. This avoids the penalty of the lengthy S-Box recomputation that other masking schemes utilize (except masking schemes with S-Box secure calculation [10, 12]). *ShiftRows* is unchanged since the underlying data is not modified. The *MixColumns*

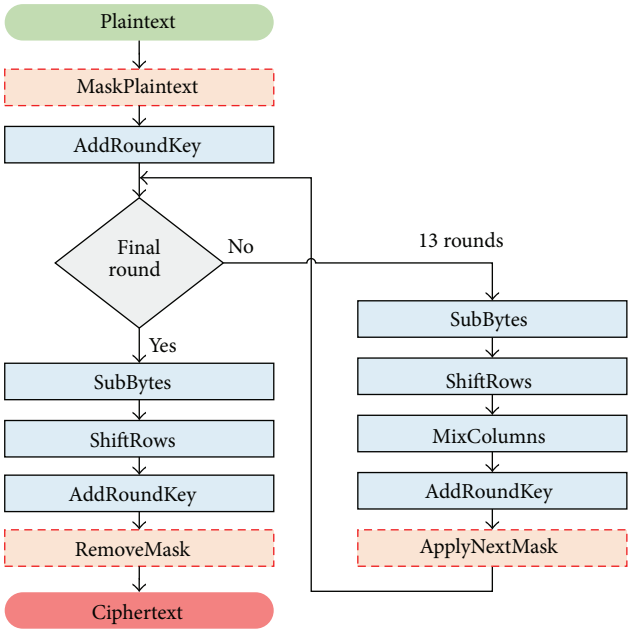


Figure 1: AES-256 with the Rotating S-Box Masking (RSM) protection. RSM is a Low Entropy Masking Scheme. The dashed boxes represent the operations added by RSM to AES.

operation is a special masked version. Afterwards, the next-round masks are applied while simultaneously removing the current-round masks, and the offset value is incremented. It is important to stress that the data never appear unmasked.

Interestingly, an optimization of RSM in terms of speed has been published in 2014 [25]. In this paper, we study the genuine RSM, as implemented in the DPA Contest V4 [22].

2.2. Power Analysis. A generic power (or EM) analysis attack has the following five steps [13]:

(1) Measure the power consumption (or EM) of a device as it encrypts (resp., decrypts) a number of plaintexts (resp., ciphertexts): we used EM traces provided by the DPA Contest V4 [22], as detailed in Section 2.3.

(2) Choose an intermediate result of the target algorithm to attack: normally, a part of the algorithm that operates on the key is attacked. However, we wish first to recover the used masks (of course, the masks set is public, but not the order in which they are used), so we target the loading of the masks, as described in Section 2.5.

(3) Calculate the intermediate results for all secret hypotheses: in this case, there are 16 possibilities for the mask set, shown in matrix **M** in Section 2.6.

(4) Apply a hypothetical power model to the calculated intermediate results: we used the Hamming weight power model, as described in Section 2.6.

(5) Compare the measured power consumption to the hypothetical power consumption to determine the secret key (or a small part of the key): this is explained in more detail in Section 2.6.

This attack is performed in two stages: (1) the preprocessing mask recovery stage and (2) CPA attack to recover the key. The basic idea is to recover an estimate of the masks from each power trace and then launch a horizontal (attacking many samples from a single trace) CPA attack against the 16 possible combinations of the mask. Recovering the masks allows us to undo the countermeasure so that we can correctly predict some intermediate value, for example, the S-Box output. Thus, a second CPA attack, vertical (attacking the same time instance across many traces) this time, reveals the key. Both stages are first-order attacks.

2.3. Experimental Setup. The AES-256 RSM is implemented on an Atmel ATMega-163 smartcard connected to a SASEBO-W board [22]. EM traces were captured using a Langer EM near-field probe RF-U 5-2, sampled at 500 MS/s by a Lecroy Waverunner 6100A oscilloscope.

2.4. Leakage Detection. In order to attack efficiently, it is important to precisely locate the leaking samples in the traces: this is the purpose of the leakage detection phase.

We use Normalized Interclass Variance (NICV) [26], which is an analysis of variance (ANOVA) F-test, to identify leakage in power traces. The NICV relies on publicly available information (such as known plaintexts or ciphertexts). Let T be the set of power traces and let X be the corresponding set of plaintext bytes. The NICV is calculated as $\mathrm{NICV} = \mathrm{Var}(\mathbf{E}[T \mid X])/\mathrm{Var}(T)$, where $\mathbf{E}$ is the expectation operator, Var is the variance operator, and $0 \leqslant |\mathrm{NICV}| \leqslant 1$. It is thus a normalized indicator of leakage, which does not require the knowledge of the key. Figure 2 shows the NICV calculated for each plaintext byte using 10,000 traces and reveals useful information to the attacker. With knowledge of the algorithm, he/she can distinguish when different operations take place. The 16 peaks in Figure 2(a) from samples 0 to 75,000 suggest the *AddRoundKey* operation, while the second set of 16 peaks beginning at sample point 10^5 signifies the *SubBytes* operation. An attacker can use this knowledge to extract leakage samples that belong to a certain operation.

The attacker now has a rough idea of the time frame when each operation takes place and can even determine the amount of time to process each byte by examining Δ, the distance between the peaks in Figure 2(b). Figure 2(a) shows that each plaintext byte is operated on only once before it enters the S-Box; that is, there is only one time interval when leakage occurs for each plaintext byte before the S-Box. Therefore, the plaintext loading, masking operation, and *AddRoundKey* must all take place within the same time interval. Moreover, the order and morphology of each NICV curve tell the attacker that the same set of operations is applied 16 times in a row, beginning with byte 0 and ending with byte 15. Consequently, the attacker now has an idea about the mask order.

2.5. Extract Leaky Samples. The attacker then chooses a window W of width Δ and extracts possible candidates for

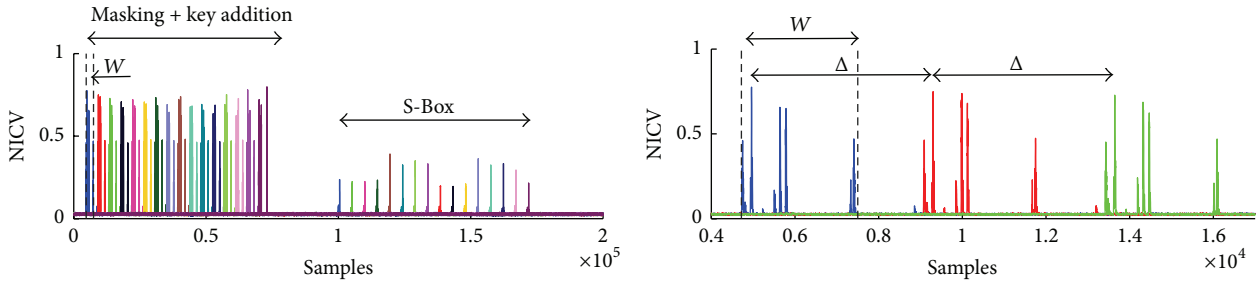


FIGURE 2: NICV for each plaintext byte over 10,000 traces. (a) AES operations are identifiable. (b) NICV for the first 3 bytes of plaintext. Each byte exhibits similar characteristics, which implies the operation taking place a number times, but each time processing different data.

the time samples when each mask is loaded. The attacker can use the NICV (or some other leakage detection tool [26] such as Sum-of-Square Differences (SOSD) or Sum-of-Square t-test (SOST)) to minimize the amount of points he/she will attack by considering only leakage measurements above a certain threshold (determined empirically), or he/she can simply attack every point in the window. The attacker selects τ samples to attack from a single power trace and stores their leakage measurements, v, into the first column of the $\tau \times 16$ matrix $\mathbf{V}$. Each column of $\mathbf{V}$ is then filled in by extracting the leakage measurement located exactly Δ samples further from the previous measurement:

$$\mathbf{V} = \begin{bmatrix} t_0 & t_0 + \Delta & t_0 + 2\Delta & \cdots & t_0 + 15\Delta \\ t_1 & t_1 + \Delta & t_1 + 2\Delta & \cdots & t_1 + 15\Delta \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ t_{\tau-1} & t_{\tau-1} + \Delta & t_{\tau-1} + 2\Delta & \cdots & t_{\tau-1} + 15\Delta \end{bmatrix}. \quad (1)$$

2.6. Recover the Mask Offset. The next step is to launch a modified CPA attack on the subtraces in $\mathbf{V}$. Since we do not know in which order the masks were loaded, we guess every combination, as shown in the 16×16 matrix $\mathbf{M} = [\mathscr{M}_0 \cdots \mathscr{M}_{15}]^\top$. Each column of $\mathbf{M}$ corresponds to an offset applied to the base set of masks $\mathscr{M}_0$, where

$$\mathbf{M} = \begin{bmatrix} m_0 & m_1 & m_2 & \cdots & m_{15} \\ m_1 & m_2 & m_3 & \cdots & m_0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ m_{15} & m_0 & m_1 & \cdots & m_{14} \end{bmatrix}. \quad (2)$$

We apply a Hamming weight power model $w_H(\cdot)$ to the mask matrix $\mathbf{M}$, which is generally a good model for microprocessors [13, 27]. The hypothetical power consumption is $\mathbf{H} = w_H(\mathbf{M})$. The next step is to compare the modeled power consumption with the measured power consumption. If we assume the power model to be linear, for example, Hamming weight or Hamming distance, a natural choice for the attack is the correlation coefficient. Correlation power analysis (CPA) evaluates the amount of correlation between a set of measured power traces T and a model of the key-dependent device leakage, L [5], and is calculated for every time sample. Pearson's correlation coefficient is calculated as $\rho(T, L) = \mathrm{cov}(T, L)/(\sigma_T \sigma_L)$; however, this can be difficult (or impossible) to compute, and so we instead use an estimate $\widehat{\rho}$ (where $|\widehat{\rho}| \leqslant 1$) which is calculated as $\sum_{i=0}^{n-1}(t_i - \overline{t}_i)(l_i - \overline{l}_i)/\sqrt{\sum_{i=0}^{n-1}(t_i - \overline{t}_i)^2 \sum_{i=0}^{n-1}(l_i - \overline{l}_i)^2}$ for the set of traces T (containing n traces t_i) and hypothetical power model L, containing n hypothetical power consumption values l. Wrong guesses for the key will have correlations close to 0, while the correct guess will have $|\widehat{\rho}|$ close to 1 (assuming the power model is accurate). We calculate $\widehat{\rho}(\mathbf{V}, \mathbf{H})$, which leads to 16 correlation coefficients. Each correlation coefficient corresponds to a mask offset. By choosing the location where $\max \widehat{\rho}(\mathbf{V}, \mathbf{H})$ occurs, we can guess the offset. The overall procedure is exhibited in Algorithm 1. Using the offset guess, we can predict the S-Box output and deploy a CPA attack to recover the key.

3. Results

This attack is feasible since the device leaks the Hamming weight of the masks when they are loaded from memory. Once the masks are recovered, extracting the key is straightforward. Our attack requires 10.1 traces to fully recover the key, while an attack on unprotected implementation requires 9.9 traces and can be considered as a lower bound regarding the number of traces. Our attack is close to that bound; the reason that we need slightly more traces is because we do not always correctly guess the offset. Comparing our offset guesses with the actual mask offsets, we were able to successfully guess the offset 91% of the time. Recall that the estimation error of the mean in a Bernoulli process is $p(1 - p)/n_{\mathrm{rep}}$, where $p = 0.91$ and n_{rep} is the number of repetitions; namely, $n_{\mathrm{rep}} = 10,000$. The success rate is estimated over 10,000 traces with accuracy $\approx 10^{-5}$.

3.1. Mask Recovery Success Rate. Figure 4(a) shows the success rate of recovering the mask for various signal-to-noise ratios (SNRs). The probability of correctly guessing the offset at random is 1/16, or 6.25%: we exceed this value for all

input: Window when masking is thought to occur W
A single power trace t
Length of masking operation Δ
Mask matrix $\mathbf{M}$
Output: The mask set $\mathcal{M}_g$ and the mask offset g
(1) $\tau \leftarrow$ ChooseSamples(W); // leakage Detection
(2) $r \leftarrow 0$; // row index for subtrace matrix V
(3) **for** $i \in \tau$ **do**
(4) **for** $j \leftarrow 0$ **to** 15 **do**
(5) $lkg \leftarrow t(i + j \cdot \Delta)$; // measured leakage at sample i for byte j
(6) $\mathbf{V}[r, j] \leftarrow lkg$; // build subtrace matrix
(7) **end**
(8) $r \leftarrow r + 1$; // increment row index
(9) **end**
(10) $\mathbf{H} \leftarrow w_H(\mathbf{M})$; // mask Hamming weight
(11) **return** $g \leftarrow \arg\max \widehat{\rho}(\mathbf{V}, \mathbf{H})$; // recover the mask offset
(12) **return** $\mathcal{M}_g \leftarrow \mathbf{M}[:, g]$; // guessed mask set

Algorithm 1: Mask recovery.

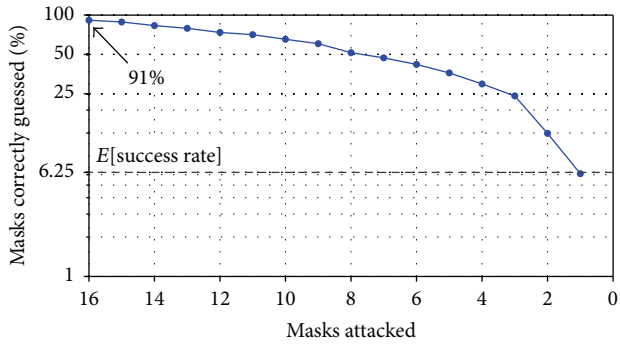


Figure 3: Mask recovery success rate as a function of number of masks attacked.

SNRs $> 2^5$ (i.e., $\sigma_{\text{noise}} > 30$). Therefore, using our method is preferred for naively guessing for most noise levels.

3.2. Tweaking the Algorithm Parameters. We examine how the algorithm parameters affect the mask recovery success rate. If only one mask (out of a possible 16) is attacked, the success rate equates to the expected value for naively guessing the mask. Indeed, with 1 mask, there is no "rotation" possible; hence, the mask is "horizontally indistinguishable." Thus, an attacker gains no advantage by trying to recover the mask by attacking only one sample, since the extra computation time does not lead to an increase in success rate. However, attacking 2 masks, that is, $\{m_0, m_1\}$, allows the pair to be distinguished with 11% success rate, slightly outperforming naive guessing. As shown in Figure 3, the success rate increases linearly as the number of masks increases, demonstrating the positive relationship between mask entropy and number of masks attacked.

The attacker can also vary the width of the window where he/she suspects the masking operation to occur. Enlarging the window linearly increases the computational effort; that is, increasing the width by n samples leads to an attack complexity of $\mathcal{O}(n)$. Compare this to a second-order attack, where an increase in n samples requires $n(n-1)/2$ calculations [28], or complexity $\mathcal{O}(n^2)$.

3.3. Comparison with State of the Art in the Presence of Noise. Noise increases the difficulty of carrying out a successful power attack; that is, an attacker is required to measure more power traces. Common sources of noise include electronic noise from other circuit components, measurement errors, and clock jitter [13, 27]. Most of the noise in cryptographic devices can be approximated by a normal distribution $\sim \mathcal{N}(0, \sigma^2)$ [13]. In order to determine the influence of noise on our attack, we artificially corrupt the power traces by introducing additive white Gaussian noise $\sim \mathcal{N}(0, \sigma^2)$.

We compare our attack with a state-of-the-art second-order attack, namely, the bivariate attack, using a centered product as combination function in [23]. This type of attack is ideal for first-order masking schemes implemented in software and was proven to be optimal in the presence of noise [23].

Figure 4(b) shows the evolution of global success rate (GSR) as a function of number of traces attacked and signal-to-noise ratio (SNR). GSR is the probability to recover the full key. We define an attack as being successful if GSR $\geqslant$ 80%; conversely, we define a failed attack if the GSR fails to reach 80% within 10,000 traces. The best-case attack scenario is SNR = 2.689; that is, no artificial noise is added. The best-case mask recovery attack requires 10 traces to succeed, whereas the best-case second-order attack does not succeed until 300 traces. The mask recovery attack is more resilient to noise since, for a given number of power traces, the success rate will be higher for all SNRs. Regardless of the noise level, our mask recovery attack (empirically) reveals the key faster than a traditional bivariate attack.

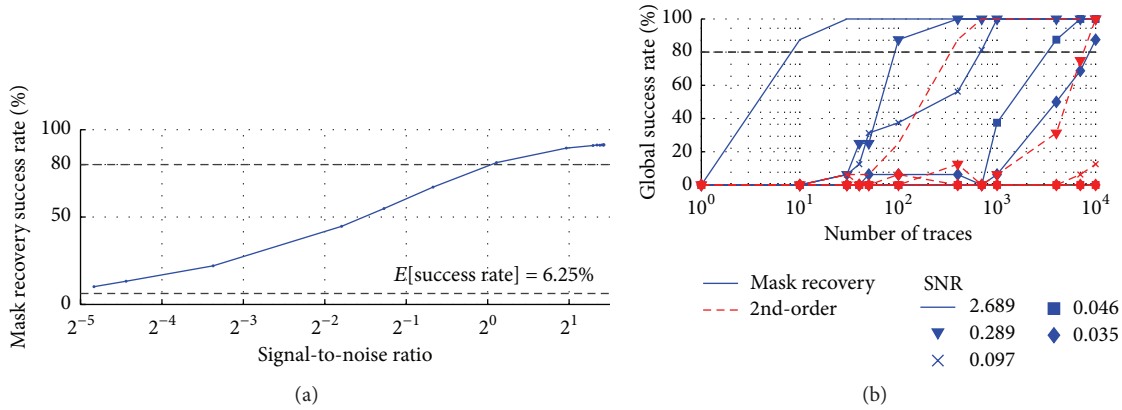


FIGURE 4: (a) Mask recovery success rate for 10,000 traces. (b) Global success rate (GSR) versus number of traces for different noise levels. The mask recovery attack outperforms the second-order attack by at least two orders of magnitude at every SNR.

The mask recovery attack outperforms the second-order attack by about two orders of magnitude for SNR $\geqslant$ 0.289. The second-order attack fails for SNR $<$ 0.289, whereas the mask recovery attack succeeds for 0.035 $\leqslant$ SNR $\leqslant$ 2.689. The lower performance of the second-order attack can be attributed to the leakage combination function. Indeed, by combining multiple leakages, the noise is amplified [23]. By choosing an optimal prediction function, the noise amplification can be minimized, but much more traces must be analyzed for a successful attack as shown in Figure 4(b).

3.4. How to Defend against this Attack? The mask set $\mathscr{M}$ is a linear code of parameters [8, 4, 4] and of weights enumerator polynomial $X^8 + 14X^4Y^4 + Y^8$, which means that one codeword has a Hamming weight of 0, another one has a Hamming weight of 8, and the remaining 14 have Hamming weights of 4. One possible solution to thwart this attack is to generate all the masks with the same Hamming weight (called constant-weight codes). In this case, every column in the hypothetical power matrix **H** would be identical. If this *constant-weight code* strategy is applied, the designer must carefully consider which masks are chosen, so that the amount of leaked information is minimized. The *constant-weight code* strategy can defend against our attack and against first-order attacks only. No set of constant-weight code masks can defend against second-order (or higher) attacks as proved in the Appendix. This only applies to 8-bit software implementation, that is, a typical smartcard; we did not consider other architectures.

The *constant-weight code* strategy assumes all bits in a computer word leak equally, which is not realistic. Thus, we propose an alternative countermeasure that requires no extra resources, defends against mask-recovery attacks, and provides the same protection against first-order attacks as plain RSM. The strategy consists in (approximately) balancing the Hamming weights of the codewords belonging to $\mathscr{M}$. It has been proven in [29] that all the cosets $y \oplus \mathscr{M}$ (for $y \in \mathbb{F}_2^8$) of the studied code $\mathscr{M}$ provide the same level of security, regarding monovariate attacks. Three options exist for the weight distribution. The probability that a randomly chosen element of the code has Hamming weight h is given below, for $h \in [\![0, 8]\!]$:

(1) $(1/16, 0, 0, 0, 14/16, 0, 0, 0, 1/16)$ if $y \in \mathscr{M}$.

(2) $(0, 1/16, 0, 7/16, 0, 7/16, 0, 1/16, 0)$ if there is one codeword of weight 1 in $y \oplus \mathscr{M}$.

(3) $(0, 0, 4/16, 0, 8/16, 0, 4/16, 0, 0)$ if there is one codeword of weight 2 in $y \oplus \mathscr{M}$.

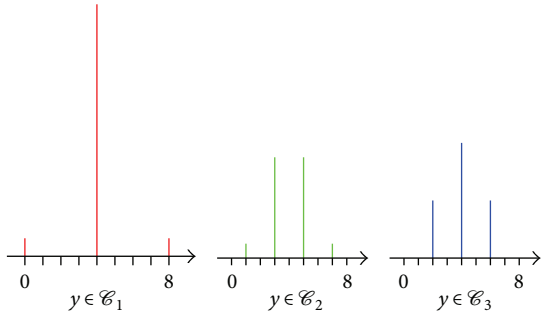
This means that $\mathbb{F}_2^8$ can be partitioned in three partitions: $\mathscr{C}_1$, $\mathscr{C}_2$, and $\mathscr{C}_3$. The distribution of $w_H(y \oplus \mathscr{M})$ is given in Figure 5, along with some noncentral moments (of degrees 1, 2, 3, and 4).

Now, by the property of the code, the variance of the Hamming weights is the same in those three cases. Namely, it is equal to 2. Indeed, the expectation of the Hamming weights is 4 in all four cases. Thus, the expectation of the square of the centered Hamming weights is, respectively, equal to

$$\begin{aligned} 2 &= \frac{1}{16} \times (-4)^2 + \frac{14}{16} \times 0 + \frac{1}{16} \times (4)^2 \\ &= \frac{1}{16} \times (-3)^2 + \frac{7}{16} \times (-1)^2 + \frac{7}{16} \times (1)^2 + \frac{1}{16} \times (3)^2 \\ &= \frac{4}{16} \times (-2)^2 + \frac{8}{16} \times 0 + \frac{4}{16} \times (2)^2 . \end{aligned} \quad (3)$$

Still, it is clear that if there is a leakage in "SPA" (Simple Power Analysis), then it is more advantageous to use the code such that the Hamming weight distribution is taking only values 2, 4, and 6. So, for instance, an improvement can be obtained by using

$$\begin{aligned} \mathscr{M}' = \{&\texttt{0x02}, \texttt{0x0d}, \texttt{0x34}, \texttt{0x3b}, \texttt{0x51}, \texttt{0x5e}, \texttt{0x67}, \\ &\texttt{0x68}, \texttt{0x97}, \texttt{0x98}, \texttt{0xa1}, \texttt{0xae}, \texttt{0xc4}, \texttt{0xcb}, \texttt{0xf2}, \\ &\texttt{0xfd}\} \end{aligned} \quad (4)$$



$\mathbb{E}[w_H^1(y \oplus \mathscr{M})]$:	64	=	64	=	64
$\mathbb{E}[w_H^2(y \oplus \mathscr{M})]$:	288	=	288	=	288
$\mathbb{E}[w_H^3(y \oplus \mathscr{M})]$:	1408	=	1408	=	1408
$\mathbb{E}[w_H^4(y \oplus \mathscr{M})]$:	*7680*	≠	*7344*	≠	*7296*

FIGURE 5: Distribution $h \in [\![0, 8]\!]$ of the Hamming weights of the cosets $y \oplus \mathscr{M}$ of the RSM code $\mathscr{M}$.

instead of

$$\mathscr{M} = \{\texttt{0x00}, \texttt{0x0f}, \texttt{0x36}, \texttt{0x39}, \texttt{0x53}, \texttt{0x5c}, \texttt{0x65}, \texttt{0x6a}, \texttt{0x95}, \texttt{0x9a}, \texttt{0xa3}, \texttt{0xac}, \texttt{0xc6}, \texttt{0xc9}, \texttt{0xf0}, \texttt{0xff}\}. \quad (5)$$

The variance of the code has not changed, only the amplitude of the patterns. Whereas the original code had a range of amplitudes from 0 to 8, the new code has a range from 2 to 6. Thus, in the presence of noise, the SNR is reduced by 50%, making it more difficult to recover the mask.

This is reflected in Figure 5 by the new proposed *affine* code $\mathscr{M}' = \mathscr{M} \oplus \texttt{0x2}$ (see (4)) having a smaller *kurtosis* (4th-degree moment) than *linear* code $\mathscr{M}$ (see (5)). Reducing the first (nonzero) correlating moment is indeed the strategy of state-of-the-art side-channel attacks on masking schemes [30].

4. Conclusion and Perspectives

We demonstrated how to recover a set of masks used in software implementation of AES with RSM. Our attack outperforms a traditional bivariate attack by two orders of magnitude and can succeed even in heavy noise. We show how the attack parameters affect the success rate; namely, attacking just 2 (out of 16) yields a better mask recovery success rate versus naive guessing. It is not enough to say implementation is first-order (or second-order, etc.) secure. Indeed, we showed that the countermeasure that could stop our attack can only defend against traditional first-order attacks. Further avenues of research involve empirically validating the countermeasure and extending this attack to other masking schemes (including higher-order masking schemes). Besides, it is interesting to study the security gain obtained by stacking other protections, such as S-Boxes shuffling, on top of RSM. Similar directions can be found in this prospective document [31] which gives the roadmap of the forthcoming DPA Contest V4 contests.

Appendix

Constant-weight codes are codes where all codewords share the same Hamming weight. They are also called m of n codes. Of particular interest are balanced codes, introduced by Knuth in 1986 [32], since they fit the basic requirement of masking. A special case for codes of length $n = 8$ is 6b/8b codes [33], used in serial communication lines to maintain DC balance in a communications system. However, in this 6b/8b code, there are 64 codewords, which is too large. Our requirements on the code can be summarized as follows: (1) the codewords must all have the same weight; (2) the code must have a large dual distance (see requirement explained in [34, 35]); (3) the code must have a size less than or equal to 16 (the number of AES substitution boxes). Nonzero balanced codes are nonlinear. Indeed, a linear code contains the null vector. Thus, the codewords have zero weight, and so the only linear balanced code is {0}.

Care must be taken that, in this appendix, "*balanced*" can have two meanings depending on the context:

> Horizontally: it can be that each codeword contains an equal number of zero and one bits.
>
> Vertically: each component (or tuples of components) is represented uniformly.

It is possible to find balanced codes with size two. For instance, on $n = 8$ bits, the code made up of codewords $(01010101)_2$ and $(10101010)_2$ is balanced. This is equivalent to saying that the code has dual distance at least 2. However, its dual distance is exactly 2: it allows protection against first-order attacks and not against zero-offset second-order attacks. The pair of two components is not balanced. For example, the two least significant bits of the codewords are $(01)_2$ and $(10)_2$: the values $(00)_2$ and $(11)_2$ are missing. In fact, a code of dual distance 3 must have at least a size 4. We need first a lemma about codes of length n with constant weight w.

Lemma A.1. *Let C be a code of length n and constant weight w. If C is balanced, then n is even and $w = n/2$.*

Proof. A balanced code is such that, for all a of unitary Hamming weight ($w_H(a) = 1$), $\sum_{c\in C}(-1)^{a\cdot c} = 0$. Now,

$$\begin{aligned}\sum_{a/w_H(a)=1}\sum_{c\in C}(-1)^{a\cdot c} &= \sum_{c\in C}\sum_{a/w_H(a)=1}(-1)^{a\cdot c}\\ &= \sum_{c\in C}\sum_{a/w_H(a)=1}(1-2(a\cdot c))\\ &= \sum_{c\in C}\left(n-2\sum_{a/w_H(a)=1}(a\cdot c)\right)\\ &= |C|(n-2w).\end{aligned} \tag{A.1}$$

This sum is also equal to zero, so we have $n = 2w$. □

We have this practically relevant result.

Proposition A.2. *A constant Hamming weight binary code has dual distance strictly less than* 3.

Proof. Let f be the indicator of a constant Hamming weight code C. We define the Fourier transform of f as $\widehat{f}(a) = \sum_x f(x)(-1)^{a\cdot x}$. If C has dual distance 3, then, for all $i, j = 1,\ldots,n$, $i \neq j$, we have $\widehat{f}(e_i \oplus e_j) = 0$, where e_i is the vector of Hamming weight 1 in which the only 1 is at coordinate i. So we have $\sum_{i=1}^{n}\widehat{f}(e_i \oplus e_j) = \widehat{f}(0) = |C|$, which is also

$$\sum_{x\in C}(-1)^{x_j}\sum_{i=1}^{n}(-1)^{x_i} = \sum_{x\in C}(-1)^{x_j}\left(n-2w_H(x)\right) = 0. \tag{A.2}$$

Therefore, C is empty. □

Proposition A.2 says that we can have constant Hamming weight codewords, but simply with protection against first-order attacks.

Example A.3 ($n = 8$ and $w = 4$). The following (nonlinear) code has parameters $(8, 16, 2)$. It has constant Hamming weight $n/2 = 4$ and is balanced (more precisely, it has dual distance 2):

$$\begin{aligned}C = \{&\texttt{0x0f}, \texttt{0xf0}, \texttt{0x33}, \texttt{0xcc}, \texttt{0x55}, \texttt{0xaa}, \texttt{0x66},\\ &\texttt{0x99}, \texttt{0xe1}, \texttt{0x1e}, \texttt{0xd2}, \texttt{0x2d}, \texttt{0xb4}, \texttt{0x4b},\\ &\texttt{0x78}, \texttt{0x87}\}.\end{aligned} \tag{A.3}$$

In order to better highlight the two dimensions of balancing of this code, we represent it in Table 1 in binary and add a "*sum of bits*" column and line.

Thus, this code can protect as well against

(i) horizontal side-channel analyses, since all codewords have the same Hamming weight (namely, $n/2 = 4$ bit),

(ii) vertical side-channel analyses, since each component is statistically balanced (i.e., each bit as probability $8/16 = 1/2$).

Table 1: Horizontally and vertically balanced code with length 8 and 16 codewords.

	Components								Sum of bits
	8	7	6	5	4	3	2	1	
Codewords	0	0	0	0	1	1	1	1	4
	1	1	1	1	0	0	0	0	4
	0	0	1	1	0	0	1	1	4
	1	1	0	0	1	1	0	0	4
	0	1	0	1	0	1	0	1	4
	1	0	1	0	1	0	1	0	4
	0	1	1	0	0	1	1	0	4
	1	0	0	1	1	0	0	1	4
	1	1	1	0	0	0	0	1	4
	0	0	0	1	1	1	1	0	4
	1	1	0	1	0	0	1	0	4
	0	0	1	0	1	1	0	1	4
	1	0	1	1	0	1	0	0	4
	0	1	0	0	1	0	1	1	4
	0	1	1	1	1	0	0	0	4
	1	0	0	0	0	1	1	1	4
Sum of bits	8	8	8	8	8	8	8	8	

Disclosure

The countermeasures described in this paper were implemented in experimental hardware and software environments. The authors of this paper have not explored the potential applicability of these countermeasures to commercially available hardware and *software*.

Disclaimer

The views expressed in this paper solely belong to the authors and do not in any way reflect the views of Intel Corporation.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

The authors are grateful to Zakaria Najm for setting up the experiments about DPA Contest V4. They also thank Guillaume Duc for managing the DPA contests. Besides, the authors are indebted to Laurent Sauvage and Jean-Luc Danger who, respectively, run the security lab and the security research group of Télécom ParisTech.

References

[1] 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, London, UK, 1999.

[2] National Institute of Standards and Technology, *FIPS 197 Advanced Encryption Standard*, National Institute of Standards and Technology, 2001.

[3] ISO, *Information Technology—Security Techniques—Encryption Algorithms—Part 3: Block Ciphers*, International Organization for Standardization, 2013.

[4] A. Kaminsky, M. Kurdziel, and S. Radziszowski, "An overview of cryptanalysis research for the advanced encryption standard," in *Proceedings of the IEEE Military Communications Conference (MILCOM '10)*, pp. 1310–1316, San Jose, Calif, USA, November 2010.

[5] É. Brier, C. Christophe, and F. Olivier, "Correlation power analysis with a leakage model," in *Cryptographic Hardware and Embedded Systems—CHES 2004*, M. Joye and Q. Jean-Jacques, Eds., vol. 3156 of *Lecture Notes in Computer Science*, pp. 16–29, Springer, Heidelberg, Germany, 2004.

[6] A. Bogdanov, "Multiple-differential side-channel collision attacks on AES," in *Cryptographic Hardware and Embedded Systems—CHES 2008*, E. Oswald and P. Rohatgi, Eds., vol. 5154 of *Lecture Notes in Computer Science*, pp. 30–44, Springer, Berlin, Germany, 2008.

[7] A. Moradi, O. Mischke, and T. Eisenbarth, "Correlation-enhanced power analysis collision attack," in *Cryptographic Hardware and Embedded Systems, CHES 2010*, S. Mangard and F.-X. Standaert, Eds., vol. 6225 of *Lecture Notes in Computer Science*, pp. 125–139, Springer, Berlin, Germany, 2010.

[8] B. Gierlichs, L. Batina, P. Tuyls, and B. Preneel, "Mutual information analysis," in *Cryptographic Hardware and Embedded Systems—CHES 2008*, E. Oswald and P. Rohatgi, Eds., vol. 5154 of *Lecture Notes in Computer Science*, pp. 426–442, Springer, Heidelberg, Germany, 2008.

[9] E. Prouff and M. Rivain, "A generic method for secure sbox implementation," in *Information Security Applications*, S. Kim, M. Yung, and H.-W. Lee, Eds., vol. 4867 of *Lecture Notes in Computer Science*, pp. 227–244, Springer, Berlin, Germany, 2007.

[10] M. Rivain and E. Prouff, "Provably secure higher-order masking of AES," in *Cryptographic Hardware and Embedded Systems, CHES 2010*, S. Mangard and F.-X. Standaert, Eds., vol. 6225 of *Lecture Notes in Computer Science*, pp. 413–427, Springer, Berlin, Germany, 2010.

[11] E. Prouff, C. Giraud, and S. Aumônier, "Provably secure S-box implementation based on fourier transform," in *Cryptographic Hardware and Embedded Systems—CHES 2006*, L. Goubin and M. Matsui, Eds., vol. 4249 of *Lecture Notes in Computer Science*, pp. 216–230, Springer, Heidelberg, Germany, 2006.

[12] J.-S. Coron, "Higher order masking of look-up tables," in *Advances in Cryptology—EUROCRYPT 2014*, H. Q. Nguyen and E. Oswald, Eds., vol. 8441 of *Lecture Notes in Computer Science*, pp. 441–458, Springer, Berlin, Germany, 2014.

[13] S. Mangard, E. Oswald, and T. Popp, *Power Analysis Attacks: Revealing the Secrets of Smart Cards*, Springer, 2008.

[14] X. Ye and T. Eisenbarth, "On the vulnerability of low entropy masking schemes," in *Proceedings of the 12th International Conference on Smart Card Research and Advanced Applications (CARDIS '13), November 2013*, A. Francillon and P. Rohatgi, Eds., vol. 8419 of *Lecture Notes in Computer Science*, pp. 44–60, Springer, Berlin, Germany, 2013.

[15] V. Grosso, F.-X. Standaert, and E. Prouff, "Low entropy masking schemes, revisited," in *Smart Card Research and Advanced Applications: 12th International Conference, CARDIS 2013, Berlin, Germany, November 27–29, 2013. Revised Selected Papers*, vol. 8419 of *Lecture Notes in Computer Science*, pp. 33–43, Springer, Berlin, Germany, 2014.

[16] C. Herbst, E. Oswald, and S. Mangard, "An AES smart card implementation resistant to power analysis attacks," in *Applied Cryptography and Network Security*, J. Zhou, M. Yung, and F. Bao, Eds., vol. 3989 of *Lecture Notes in Computer Science*, pp. 239–252, Springer, Berlin, Germany, 2006.

[17] M. Nassar, Y. Souissi, S. Guilley, and J.-L. Danger, "RSM: a small and fast countermeasure for AES, secure against 1st and 2nd-order zero-offset SCAs," in *Proceedings of the 15th Design, Automation and Test in Europe Conference and Exhibition (DATE '12)*, pp. 1173–1178, Dresden, Germany, March 2012.

[18] C. Clavier, B. Feix, G. Gagnerot, M. Roussellet, and V. Verneuil, "Horizontal correlation analysis on exponentiation," in *Information and Communications Security: 12th International Conference, ICICS 2010, Barcelona, Spain, December 15–17, 2010. Proceedings*, vol. 6476 of *Lecture Notes in Computer Science*, pp. 46–61, Springer, Berlin, Germany, 2010.

[19] J. Pan, J. I. den Hartog, and J. Lu, "You cannot hide behind the mask: power analysis on a provably secure S-box implementation," in *Information Security Applications*, H. Y. Youm and M. Yung, Eds., vol. 5932 of *Lecture Notes in Computer Science*, pp. 178–192, Springer, Berlin, Germany, 2009.

[20] M. Tunstall, C. Whitnall, and E. Oswald, "Masking tables—an underestimated security risk," in *Fast Software Encryption: 20th International Workshop, FSE 2013, Singapore, March 11–13, 2013. Revised Selected Papers*, vol. 8424 of *Lecture Notes in Computer Science*, pp. 425–444, Springer, Berlin, Germany, 2014.

[21] N. Bruneau, S. Guilley, Z. Najm, and Y. Teglia, "Multi-variate high-order attacks of shuffled tables recomputation," in *Cryptographic Hardware and Embedded Systems—CHES 2015: 17th International Workshop, Saint-Malo, France, September 13–16, 2015. Proceedings*, H. Handschuh and T. Güneysu, Eds., vol. 9293 of *Lecture Notes in Computer Science*, Springer, Berlin, Germany, 2015.

[22] DPA Contest, 2014, http://www.dpacontest.org.

[23] E. Prouff, M. Rivain, and R. Bevan, "Statistical analysis of second order differential power analysis," *IEEE Transactions on Computers*, vol. 58, no. 6, pp. 799–811, 2009.

[24] A. DeTrano, S. Guilley, X. Guo, N. Karimi, and R. Karri, "Exploiting small leakages in masks to turn a second-order attack into a first-order attack," in *Proceedings of the 4th Workshop on Hardware and Architectural Support for Security and Privacy (HASP '15)*, pp. 7:1–7:5, ACM, Minneapolis, Minn, USA, June 2015.

[25] N. Yamashita, K. Minematsu, T. Okamura, and Y. Tsunoo, "A smaller and faster variant of RSM," in *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE '14)*, pp. 1–6, IEEE, Dresden, Germany, March 2014.

[26] S. Bhasin, J.-L. Danger, S. Guilley, and Z. Najm, "Side-channel leakage and trace compression using normalized interclass variance," in *Proceedings of the 3rd Workshop on Hardware and Architectural Support for Security and Privacy (HASP '14)*, pp. 7:1–7:9, ACM, Minneapolis, Minn, USA, June 2014.

[27] P. Kocher, J. Jaffe, B. Jun, and P. Rohatgi, "Introduction to differential power analysis," *Journal of Cryptographic Engineering*, vol. 1, no. 1, pp. 5–27, 2011.

[28] E. Oswald, S. Mangard, C. Herbst, and S. Tillich, "Practical second-order DPA attacks for masked smart card implementations of block ciphers," in *Topics in Cryptology—CT-RSA 2006*, vol. 3860 of *Lecture Notes in Computer Science*, pp. 192–207, Springer, Berlin, Germany, 2006.

[29] C. Carlet and S. Guilley, "Side-channel indistinguishability," in *Proceedings of the 2nd International Workshop on Hardware and Architectural Support for Security and Privacy (HASP '13)*, pp. 9:1–9:8, ACM, Tel Aviv, Israel, June 2013, https://hal.archives-ouvertes.fr/hal-00826618.

[30] A. Moradi and F.-X. Standaert, "Moments-correlating DPA," IACR Cryptology ePrint Archive 2014:409, 2014.

[31] S. Bhasin, N. Bruneau, J.-L. Danger, S. Guilley, and Z. Najm, "Analysis and improvements of the DPA contest v4 implementation," in *Security, Privacy, and Applied Cryptography Engineering: 4th International Conference, SPACE 2014, Pune, India, October 18–22, 2014. Proceedings*, R. S. Chakraborty, V. Matyas, and P. Schaumont, Eds., vol. 8804 of *Lecture Notes in Computer Science*, pp. 201–218, Springer, Berlin, Germany, 2014.

[32] D. E. Knuth, "Efficient balanced codes," *IEEE Transactions on Information Theory*, vol. 32, no. 1, pp. 51–53, 1986.

[33] X. A. Widmer, "DC-balanced 6b/8b transmission code with local parity," US Patent 6,876,315, 2005.

[34] S. Bhasin, C. Carlet, and S. Guilley, "Theory of masking with codewords in hardware: low-weight dth-order correlation-immune boolean functions," 2013, http://eprint.iacr.org/2013/303.

[35] C. Carlet and S. Guilley, "Correlation-immune Boolean functions for easing counter measures to side-channel attacks," in *Algebraic Curves and Finite Fields: Cryptography and Other Applications*, vol. 16 of *Radon Series on Computational and Applied Mathematics*, chapter 3, pp. 41–70, De Gruyter, Berlin, Germany, 2014.