Analysis and Improvement of Differential Computation Attacks against Internally-Encoded White-Box Implementations

Matthieu Rivain[1] and Junwei Wang[1,2,3]

[1] CryptoExperts
[2] University of Luxembourg
[3] University Paris 8
{matthieu.rivain,junwei.wang}@cryptoexperts.com

Abstract. *White-box cryptography* is the last security barrier for a cryptographic software implementation deployed in an untrusted environment. The principle of *internal encodings* is a commonly used white-box technique to protect block cipher implementations. It consists in representing an implementation as a network of look-up tables which are then *encoded* using randomly generated bijections (the internal encodings). When this approach is implemented based on *nibble* (*i.e.* 4-bit wide) encodings, the protected implementation has been shown to be vulnerable to *differential computation analysis* (DCA). The latter is essentially an adaptation of differential power analysis techniques to *computation traces* consisting of runtime information, *e.g.*, memory accesses, of the target software. In order to thwart DCA, it has then been suggested to use wider encodings, and in particular *byte* encodings, at least to protect the outer rounds of the block cipher which are the prime targets of DCA.

In this work, we provide an in-depth analysis of when and why DCA works. We pinpoint the properties of the target variables and the encodings that make the attack (in)feasible. In particular, we show that DCA can break encodings wider than 4-bit, such as *byte* encodings. Additionally, we propose new DCA-like attacks inspired from side-channel analysis techniques. Specifically, we describe a *collision attack* particularly effective against the internal encoding countermeasure. We also investigate *mutual information analysis* (MIA) which naturally applies in this context. Compared to the original DCA, these attacks are also passive and they require very limited knowledge of the attacked implementation, but they achieve significant improvements in terms of trace complexity. All the analyses of our work are experimentally backed up with various attack simulation results. We also verified the practicability of our analyses and attack techniques against a publicly available white-box AES implementation protected with byte encodings –which DCA has failed to break before– and against a "masked" white-box AES implementation –which intends to resist DCA.

Keywords: White-box Cryptography · Internal Encoding · Differential Computation Analysis · Collision Attack · Mutual Information Analysis

1 Introduction

Software implementations of cryptographic algorithms in the real world suffer more severe challenges than expected in their design model. In addition to the well-known *side-channel analysis* (SCA) attacks [Koc96, KJJ99, Cor99, PQ03], an adversary sometimes might gain full access to a software implementation of a cryptographic algorithm. She could then try to extract the underlying secret key by all kinds of means, *e.g.* by performing static or

IACR Transactions on Cryptographic Hardware and Embedded Systems ISSN 2569-2925,
Vol. 2019, No. 2, pp. 225–255
DOI:10.13154/tches.v2019.i2.225-255

dynamic analysis of the controlled binary [Sv99], or by interfering in the execution and exploiting leakage from erroneous outputs as in *(differential) fault analysis* [BDL97, BS97].

The seminal work on *white-box cryptography* (WBC), introduced by Chow *et al.* in 2002 [CEJvO02a] intends to protect cryptographic software against these kinds of threats. In particular, it aims to render *key extraction* difficult –if not infeasible– to any malicious party that would gain full access to the program and/or the execution environment. With the development of smartphones and wearable devices embedding third party applications, more and more cryptographic implementations are being deployed in untrusted environments, resulting in a growing interest for white-box cryptography.

Despite its practical interest, no provably secure white-box implementation can be found in the literature after almost 20 years of exploration. Many different white-box schemes have been proposed to protect implementations of block ciphers [CEJvO02a, CEJvO02b, BCD06, XL09, Kar11] but all these solutions have been broken by structural attacks [BGEC04, GMQ07, MGH09, MRP13, LRM$^+$14]. This situation has pushed the industry to deploy *home-made* white-box implementations, the designs of which are kept secret, to meet the growing needs. Although these implementations might not be secure against a well-informed adversary, the security of their designs can make them practically hard to break since *e.g.* known structural attacks do not apply as is.

The *internal encoding* principle was first put forward in the seminal white-box paper [CEJvO02a] and is still commonly used as a *countermeasure* to provide some levels of protection in a white-box context. The main principle is first to turn the implementation with some key into a sequence of connected look-up tables, then a bijection and its inverse are applied to each pair of connected tables to hide their content. Encodings can be divided into two categories: *internal* and *external* encodings. In particular, external encodings are bijections applied on the input or output of the cipher. However, the application of external encodings changes the specification of the original cipher, which is prohibitive for many use cases of white-box cryptography based on predefined (standard) cryptographic algorithms. For this reason, we only focus on internal encodings in the present paper.

At CHES 2016, Bos *et al.* proposed to use *differential computation analysis* (DCA) to attack white-box implementations [BHMT16]. DCA is mainly an adaptation of the *differential power analysis* (DPA) techniques [KJJ99] to the white-box context. It exploits the fact that the variables appearing in the computation in some unknown encoded form might have a strong linear correlation with the original plain values. It works by first collecting some *computation traces*, which are composed of the runtime computed values over several executions through a dynamic instrumentation tool, such as Intel PIN. One then makes a key guess and predicts the value of a (supposedly) computed intermediate variable, and compute the correlation between this prediction and each sample of the computation trace. The key guess with the highest peak in the obtained correlation trace is selected as key candidate. The power of DCA comes from the fact that the attacker does not need to know the underlying implementation details. This approach has been shown especially effective to break many publicly available white-box implementations [BHMT16] (many of which are protected with internal encodings), and it was extensively used as a white-box cryptanalytic technique in the recent *WhibOx* contest.[1]

Related Works. Although impressive, the effectiveness of DCA to break implementations protected by internal encodings was left without formal explanation in [BHMT16]. This gap was addressed recently by Bock *et al.* who provide in [BBMT18, BBB$^+$17] a first formal explanation of the DCA success. However their analysis is partly experimental (in particular for wrong key guesses) and it is limited to the case of linear and/or nibble encodings, which appeals for a more formal and more general analysis. DCA was also recently extended by Bogdanov *et al.* in [BRVW18] to defeat implementations protected

[1] See `https://whibox-contest.github.io`.

by standard side-channel countermeasures such as *masking* and *shuffling*. The authors propose a thorough analysis of this setting but they do not consider the use of internal encodings which is a common countermeasure in the white-box context.

Our Contributions. This paper has three main contributions:

1. **Analysis and improvement of DCA.** We first reformulate DCA in the well-established theory of Boolean functions. We provide an in-depth analysis of the attack that pinpoints when and why DCA works against encoded implementations. Our results include close formulas for the DCA success probability with respect to different parameters (and in particular the encoding width). This allows us to validate several formal and informal claims of [BBMT18]. Moreover, we show that DCA can actually break byte (and wider) encodings by targeting variables beyond the first round of the cipher.

2. **New DCA-like collision attack.** We propose a new kind of collision attack in the passive white-box setting where an adversary observes a computation trace with only limited knowledge of the underlying implementation. Our attack is as generic as DCA but it can defeat internal encodings with a significantly lower trace complexity. For instance, we could break a publicly available implementation protected with byte encodings in about 60 traces with our collision attack whereas DCA requires about 1800 traces. We give some theoretical analysis of our collision attack and show that its success can be formulated as a *balls-and-containers* game.

3. **Application of mutual information analysis (MIA).** We suggest to apply MIA to the passive white-box attack setting. In particular, we propose a more efficient variant of MIA attack against internal encodings. We also analyze the deep connection between this improved MIA and our collision attack.

All our analyses are backed up with attack simulations and practical attack experiments against a publicly available white-box implementation protected with byte encodings –which DCA has failed to break before– and against a "masked" white-box AES implementation –which intends to resist DCA.

Organization. We first introduce the notations and the preliminaries in Section 2. Section 3 reviews the internal encoding countermeasure and the passive white-box attack model considered in this article. Afterwards, the three different attack techniques, namely, DCA, collision attack, and MIA are introduced and analyzed in Section 4, Section 5 and Section 6 respectively. Some comparison and conclusion are finally given in Section 7 and Section 8.

2 Preliminaries

Along this paper, we use the following notations. The random variables are denoted by uppercase Latin letters, *e.g.*, X, while the lowercase letter x denotes a particular realization of X. We further denote vectors by bold symbols, *e.g.*, $\boldsymbol{x}$. Latin letters in calligraphic format, *e.g.* $\mathcal{X}$ are used to denote random distributions and finite sets. For a random variable $X \sim \mathcal{X}$, Φ_X or $\Phi_{\mathcal{X}}$ denotes the cumulative distribution function (CDF) of $\mathcal{X}$ and $\Pr_{\mathcal{X}}(x)$ or $\Pr_X(x)$ denotes the probability mass function (PMF) of $\mathcal{X}$ evaluated on x. Finally, we denote $[n]$ the set of positive integers not greater than n, *i.e.*, $\{1, 2, \cdots, n\}$.

2.1 Hypergeometric Distribution

The *hypergeometric distribution* $\mathcal{HG}(\alpha, \beta, \tau)$ is a discrete distribution describing the probability of the number of successes in τ draws without replacement where the sample population has size β and contains exactly α successes. The PMF of $\mathcal{HG}(\alpha, \beta, \tau)$ is

$$\Pr_{\mathcal{HG}(\alpha,\beta,\tau)}(t) = \frac{\binom{\alpha}{t}\binom{\beta-\alpha}{\tau-t}}{\binom{\beta}{\tau}} .$$

And its variance is $\mathrm{Var}\big(\mathcal{HG}(\alpha,\beta,\tau)\big) = \tau\frac{\alpha}{\beta}\Big(1-\frac{\alpha}{\beta}\Big)\Big(\frac{\beta-\tau}{\beta-1}\Big)$.

In this paper, we will consider a special case of hypergeometric distribution, denoted $\widetilde{\mathcal{HG}}(n)$, which is defined as

$$\widetilde{\mathcal{HG}}(n) = \mathcal{HG}\big(2^{n-1}, 2^n, 2^{n-1}\big) ,$$

for some $n \in \mathbb{N}$. The variance of this distribution satisfies $\mathrm{Var}\big(\widetilde{\mathcal{HG}}(n)\big) = \frac{2^{2n-4}}{2^n-1}$.

2.2 Pearson's Correlation Coefficient

The *Pearson's correlation coefficient* is a measure of the *linear* correlation between two random variables X and Y. It is defined by the following equation

$$\mathrm{Cor}(X,Y) = \frac{\mathrm{Cov}(X,Y)}{\sigma_X \cdot \sigma_Y} = \frac{\mathrm{E}(XY) - \mathrm{E}(X)\mathrm{E}(Y)}{\sqrt{\mathrm{E}(X^2) - (\mathrm{E}(X))^2}\sqrt{\mathrm{E}(Y^2) - (\mathrm{E}(Y))^2}} ,$$

where $\mathrm{Cov}(X,Y)$ is the *covariance* between X and Y, $\mathrm{E}(X)$ is the *expectation* of X, and σ_X is the *standard deviation* of X. The correlation coefficient satisfies $-1 \le \mathrm{Cor}(X,Y) \le 1$, where the lower bound is reached when X and Y are negatively linearly correlated, and the upper bound is achieved when X and Y are positively linearly correlated. A zero correlation is obtained if X and Y are *linearly* independent (which doesn't imply that X and Y are independent).

2.3 Boolean Functions

Let $\mathbb{F}_2$ denote the field with 2 elements and let n be a positive integer. A *Boolean function* f with n variables is a function from $\mathbb{F}_2^n$ to $\mathbb{F}_2$.

The *weight* of a Boolean function f, denoted by $\mathsf{wt}(f)$, is the number of 1s in its value table, *i.e.* $\mathsf{wt}(f) = |\{x \in \mathbb{F}_2^n : f(x) = 1\}|$. A Boolean function f is *balanced* if $\mathsf{wt}(f) = 2^{n-1}$ (*i.e.* if it has as many 0 outputs as 1 outputs). The set of balanced n-variable Boolean functions is denoted by $\mathcal{B}(n)$ in this paper. The *bias* (or *imbalance*) of a Boolean function f is defined as

$$\mathrm{B}(f) = \sum_{x\in\mathbb{F}_2^n} (-1)^{f(x)} = 2^n - 2 \cdot \mathsf{wt}(f) . \quad (1)$$

We have $\mathrm{B}(1+f) = -\mathrm{B}(f)$ and $\mathrm{B}(f) = 0$ iff $f \in \mathcal{B}(n)$.

A Boolean function f has a unique *algebraic normal form* (ANF), which is given by a set of coefficients $a_u \in \mathbb{F}_2$, $u \in \{0,1\}^n$ as $f(x_1, x_2, \cdots, x_n) = \sum_{u\in\{0,1\}^n} a_u x^u$ where $x^u = \prod_{i=1}^n x_i^{u_i}$.

2.4 Boolean Correlation

Let f, g be two n-variable Boolean functions and $b, b_1, b_2 \in \mathbb{F}_2$, define

$$N_b^f = \Big|\big\{x \in \mathbb{F}_2^n : f(x) = b\big\}\Big| ,$$

$$N_{b_1 b_2}^{f,g} = \Big|\big\{x \in \mathbb{F}_2^n : f(x) = b_1, g(x) = b_2\big\}\Big| .$$

Then the Pearson's correlation between $f(X)$ and $g(X)$ for a uniform random input X over $\mathbb{F}_2^n$, simply denoted by $\mathrm{Cor}(f,g)$ for the sake of clarity, satisfies

$$\mathrm{Cor}(f,g) = \frac{N_{11}^{f,g} N_{00}^{f,g} - N_{10}^{f,g} N_{01}^{f,g}}{\sqrt{N_1^f N_0^f N_1^g N_0^g}} \ ,$$

If $f, g \in \mathcal{B}(n)$, the above can be simplified to

$$\mathrm{Cor}(f,g) = \frac{1}{2^n} \sum_{x \in \mathbb{F}_2^n} (-1)^{f(x)+g(x)} = \frac{1}{2^n} \mathrm{B}(f+g) \ .$$

2.5 Vectorial Boolean Functions

Let n, m be two positive integers. A (n,m)-*vectorial Boolean function* (VBF) f is a function from $\mathbb{F}_2^n$ to $\mathbb{F}_2^m$. A VBF is balanced if the cardinality of $\{x \in \mathbb{F}_2^n : f(x) = y\}$ equals 2^{n-m} for every $y \in \mathbb{F}_2^m$. Given a (n,m)-VBF f, the Boolean functions $f_1, f_2, \cdots, f_m$ such that $f(x) = \big(f_1(x), f_2(x), \cdots, f_m(x)\big)$, are called the *coordinatefunctions* of f. If a (n,m)-VBF is balanced, then any non-zero linear combination of its coordinate functions is balanced.

3 Internal Encodings and Adversary Model

3.1 Internal Encoding

The principle of internal encodings was proposed by Chow *et al.* at SAC 2002 [CEJvO02a] to protect block ciphers from key extraction in the white-box context. The fundamental idea is first to turn the implementation into a network of look-up tables with a hard-coded key. Then these tables are encoded by randomly sampled bijections, called *internal encodings* in the literature.[2] More specifically, for any pair of connected tables, an invertible transformation T is applied to the output of the first table, and then the inverse of T is applied to the input of the subsequent table. A comprehensive tutorial of the internally-encoded AES implementation of Chow *et al.* can be found in [Mui13].

For a given encoded implementation of some block cipher, any intermediate variable can be expressed as the output of an internal encoding $\varepsilon : \mathbb{F}_2^m \mapsto \mathbb{F}_2^m$ applied to some variable s. The latter can be expressed through a key dependent function $\varphi_k : \mathbb{F}_2^n \mapsto \mathbb{F}_2^m$ of a public variable x (part of the plaintext or the ciphertext) for some subkey $k \in \mathcal{K}$. This formalism is depicted in Figure 1. In this paper, we consider that φ_k is a balanced VBF which shall be the case for a vast majority of attack scenarios. In practice, the bit-size m of the internal encodings is usually small in consideration of the code size (since storage is exponential in m). For instance, the AES implementation of Chow *et al.* is based on nibble encodings (*i.e.* $m = 4$).

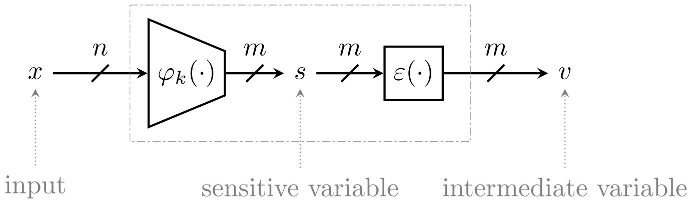

Figure 1: An illustration of how a sensitive intermediate variable is encoded.

[2] As aforementioned, the application of external encodings is out of the scope of this work.

Whenever the target variable is such that $n = m$, namely if the number of plaintext bits in its expression equals the bit-size of the encoding, and assuming that ε is a uniformly sampled bijection, then $\varepsilon \circ \varphi_k$ is a uniform random bijection which is independent of the secret key k. Consequently, there is no leakage of the underlying key k from the table itself, nor from the encoded variable $\varepsilon(s)$. This makes appear a fundamental requirement common to all kinds of DCA-like attack against encoded implementations: the target intermediate variable must be a key-dependent *non-injection*, *i.e.*, φ_k must be such that $n > m$.

Note that although the above requirement is mandatory for a DCA-like attack to work, it does not represent a strong constraint for the attacker since such key-dependent non-injective intermediate variables naturally exist in standard block cipher designs. Indeed, for security reasons, a block cipher should have a good *diffusion*, which means that each bit of the internal state should depend on all the bits of the plaintext after a few rounds. For most block ciphers, such key-dependent non-injective variables can be found in the first couple of rounds, *e.g.* in AES 1st round, a nibble of an S-box output or a byte of the MixColumn output. The former is the typical target of DCA attacks in the literature [BHMT16, BBMT18], while the latter will be the case study in our experiments. Note that any byte of the AES state in the second (or a later) round is also a key-dependent non-injective variable.

3.2 Passive Adversary Model

We consider a passive adversary who invokes the white-box implementation many times by using freely chosen inputs. With the help of *dynamic binary instrumentation* (DBI) tools, she can record the so-called *computation trace*, consisting of the accessed (read and write) memory addresses, values and associated instructions. In particular, the table lookups are included in the computation traces. Each computation trace $\boldsymbol{v}$ is composed of T samples, *i.e.*,

$$\boldsymbol{v} = (v_1, v_2, \cdots, v_T),$$

where $v_j \in \mathcal{V}$, for every $j \in [T]$ for some set $\mathcal{V}$. In the following, we consider an adversary that collects N computation traces $\left(\boldsymbol{v}^{(1)}, \boldsymbol{v}^{(2)}, \cdots, \boldsymbol{v}^{(N)}\right)$ corresponding to N inputs $\left(x^{(1)}, x^{(2)}, \cdots, x^{(N)}\right)$ of the target variable. Alternatively, the N traces can be interpreted as an $N \times T$ matrix, where a row is a computation trace $\boldsymbol{v}^{(i)}$, and a column is composed of N instances of the same intermediate variable over different computations.

In some attack scenarios, the adversary may first preprocess the traces before launching her analysis. For instance, she can remove all the constant or duplicate samples in the traces; she can also split each multi-bit sample into a tuple of bits to get a *binary* trace in which $\mathcal{V} = \{0, 1\}$.

The adversary attempts to build a *distinguisher* D, mapping the inputs $\left(x^{(i)}\right)_i$ and the corresponding computation traces $\left(\boldsymbol{v}^{(i)}\right)_i$ to a *score vector*:

$$(\delta_k)_{k \in \mathcal{K}} = \mathsf{D}\Big(\big(x^{(1)}, \cdots, x^{(N)}\big), \big(\boldsymbol{v}^{(1)}, \cdots, \boldsymbol{v}^{(N)}\big)\Big).$$

A distinguisher can be built with outputs in a similar way. Without loss of generality, we only consider the distinguisher built with inputs. The adversary then selects the key guess k with the highest score δ_k as the candidate for the correct key value k^*. Hence, the success probability of the attack is defined as

$$p_{\text{succ}} = \Pr\Big(k^* = \operatorname*{argmax}_{k \in \mathcal{K}} \delta_k\Big) \ .$$

In a typical scenario, the adversary targets a key dependent sensitive variable $s = \varphi_k(x)$. Then for each key guess $k \in \mathcal{K}$, she computes a prediction of the target sensitive variable

$\varphi_k(x)$. The score for a key guess $k \in \mathcal{K}$ is calculated by measuring some form of dependency between the predictions $\varphi_k\left(x^{(i)}\right)$ and the computation traces $\boldsymbol{v}^{(i)}$.

We consider three different distinguishers in this work. Specifically, we first provide an in-depth analysis of the DCA distinguisher [BHMT16]. Then we suggest to use a new DCA-like collision attack to analyze white-box implementations. We further demonstrate the application of MIA in this setting and improve it for attacking internal encodings.

4 Differential Computation Analysis

Differential computation analysis, proposed at CHES 2016, breaks a massive amount of publicly accessible white-box implementations [BHMT16]. DCA simply consists in applying *differential power analysis* (DPA) techniques to computation traces. In this section, we first formally define the DCA distinguisher in Section 4.1. Then we provide an in-depth analysis of DCA against implementations protected by internal encodings in Section 4.2. Specifically, we pinpoint when and why DCA works by using some properties of Boolean functions. In particular, we show that DCA can break encodings wider than 4 bits (such as byte encodings) and that it can target variables deeper in the cipher than in the first (or last) round. We validate our theoretical analysis through several simulations in Section 4.3 and successful attacks on two white-box implementations in Section 4.4. In the end, several specific discussions, including a comparison with [BBMT18], are conducted.

4.1 DCA Distinguisher

In the following, we denote respectively by k a key guess, k^* the correct key guess, and $k^\times$ a wrong key guess. For clarity, we abuse notations by skipping the parameter k in the selection function by letting $\varphi = \varphi_k$, $\varphi^* = \varphi_{k^*}$ and $\varphi^\times = \varphi_{k^\times}$. We then denote φ_i the i-th coordinate of φ, and v_j the j-th sample in a trace.

The DCA distinguisher for a key guess k is calculated as the maximal absolute value of the correlation between the i-th bit of hypothesized sensitive variable $\varphi_i(X)$, for some $i \in [m]$, and each trace sample V_j, that is

$$\delta_k^{\text{dca}} = \max_{j \in [T]} \left|\operatorname{Cor}\bigl(\varphi_i(X), V_j\bigr)\right| .$$

In practice, the adversary does not compute the exact value of the above correlation, but an estimation of it based on sampled computation traces corresponding to random plaintexts. Moreover she could try several different selection functions φ and any $i \in [m]$ until the correlation for one key guess can be distinguished from the others. We note that this definition of DCA distinguisher is similar to [BRVW18].

Although DCA can work whatever the definition space $\mathcal{V}$ of the samples in the computation trace, we consider hereafter that all the samples have been previously split into bits before computing the correlation scores (*i.e.* we have $\mathcal{V} = \{0, 1\}$).

4.2 Analysis of DCA against Encoded Implementations

In the following, we first introduce the formal (idealized) model which we use for our theoretical analysis. We then exhibit the distributions of the underlying correlation scores for different key guesses, and we analyze the success rate of DCA in this model.

Idealized model. We perform our analysis in an idealized model in which the functions $(\varphi_k)_{k \in \mathcal{K}}$ are modeled as independent random balanced (n, m)-VBF. Using such an ideal assumption is common in symmetric cryptanalysis and it is justified in practice since the S-boxes are usually chosen in such a way that φ_k and $\varphi_{k'}$ are highly uncorrelated (as independent random functions would be). To get a formal model for the full computation trace, we further ideally assume that except the m coordinates of $\varepsilon \circ \varphi_{k^*}(X)$, the samples can be expressed as $V_j = f_j(X)$ where the f_j's are uniform random functions of $\mathcal{B}(n)$.

Note that this idealized model is used for our theoretical analysis which is then challenged and validated using attack simulations and practical attack experiments.

Distributions of correlation scores. Hereafter, we characterize the correlation score $\mathrm{Cor}\big(\varphi_i(X), V_j\big)$ when V is the target encoded variable *i.e.* $V = \varepsilon \circ \varphi^*(X)$ for a uniformly distributed plaintext variable X and a random m-bit encoding ε. According to our model, we have $\varphi = \varphi^*$ if the key guess is correct (*i.e.* $k = k^*$); φ and φ^* are mutually independent otherwise. We have:

$$\mathrm{Cor}\big(\varphi_i(X), V_j\big) = \mathrm{Cor}(\varphi_i, \varepsilon_j \circ \varphi^*) = \frac{1}{2^n} \cdot \mathrm{B}(\varepsilon_j \circ \varphi^* + \varphi_i).$$

Our analysis is based on the following key lemma.

Lemma 1. *Let $g \in \mathcal{B}(n)$. Let f be a random function uniformly sampled in $\mathcal{B}(n)$ independently of g. Then we have*

$$B(f+g) = 4 \cdot N_{00}^{f,g} - 2^n \quad \textit{with} \quad N_{00}^{f,g} \sim \widetilde{\mathcal{HG}}(n) \ ,$$

where $\widetilde{\mathcal{HG}}(n)$ is the hypergeometric distribution with parameters $(2^{n-1}, 2^n, 2^{n-1})$.

Proof. By Equation 1, we have $\mathrm{B}(f+g) = 2^n - 2 \cdot \mathsf{wt}(f+g)$. Since both f and g are balanced, we have $\mathsf{wt}(f+g) = 2^n - 2 \cdot N_{00}^{f,g}$ (see definition of $N_{00}^{f,g}$ in Section 2.4), which implies $\mathrm{B}(f+g) = 4 \cdot N_{00}^{f,g} - 2^n$. Since $N_{00}^{f,g}$ is the number of inputs x for which $f(x) = 0$ among the 2^{n-1} inputs satisfying $g(x) = 0$, we directly get $N_{00}^{f,g} \sim \widetilde{\mathcal{HG}}(n)$ by definition of the hypergeometric distribution and the uniformity of f. □

For the sake of clarity, we let Y_k be the bias $\mathrm{B}(\varepsilon_j \circ \varphi^* + \varphi_i)$ for a key guess $k \in \mathcal{K}$, and only consider Y_k in our analysis. For the correct key guess k^*, we have

$$Y_{k^*} = \mathrm{B}(\varepsilon_j \circ \varphi^* + \varphi_i^*) = 2^{n-m} \cdot \mathrm{B}(\varepsilon_j + l_i) \ ,$$

where $l_i(x) = x_i$ (the ith coordinate of x). Since ε is an m-bit random permutation, ε_j is randomly distributed over $\mathcal{B}(m)$. According to Lemma 1, we then get

$$Y_{k^*} = 2^{n-m+2} \cdot N_{00}^{\varepsilon_j, l_i} - 2^n \quad \text{with} \quad N_{00}^{\varepsilon_j, l_i} \sim \widetilde{\mathcal{HG}}(m) \ .$$

On the other hand, for an incorrect key guess $k^\times \in \mathcal{K} \setminus \{k^*\}$, we have –according to our ideal assumption– that $\varepsilon_j \circ \varphi^*$ and $\varphi_j^\times$ are randomly and independently distributed over $\mathcal{B}(n)$, which implies

$$Y_{k^\times} = \mathrm{B}(\varepsilon_j \circ \varphi^* + \varphi_i^\times) = 4 \cdot N_{00}^{\varepsilon_j \circ \varphi, \varphi_i^\times} - 2^n \quad \text{with} \quad N_{00}^{\varepsilon_j \circ \varphi, \varphi_i^\times} \sim \widetilde{\mathcal{HG}}(n) \ . \tag{2}$$

The mean of Y_{k^*} and $Y_{k^\times}$ are both 0, but recalling that $\mathrm{Var}\big(\widetilde{\mathcal{HG}}(\ell)\big) = \frac{2^{2\ell-4}}{2^\ell - 1}$ (which equally holds for $\ell = n$ or m), their variances satisfy

$$\mathrm{Var}(Y_{k^*}) = \frac{2^{2n}}{2^m - 1} \quad \text{and} \quad \mathrm{Var}(Y_{k^\times}) = \frac{2^{2n}}{2^n - 1} \ . \tag{3}$$

This makes the distributions of Y_{k^*} and $Y_{k^\times}$ easily distinguishable for practical parameters m and n (with $n > m$) as illustrated hereafter.

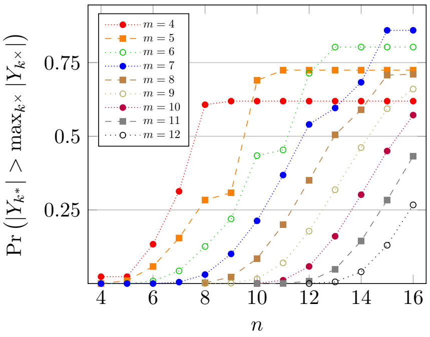

Figure 2: $\Pr\big(|Y_{k^*}| > \max_{k^\times} |Y_{k^\times}|\big)$ for $n \in \{4, \cdots, 16\}$ and $m \in \{4, \cdots, \max(n, 12)\}$ and $|\mathcal{K}| = 2^n$.

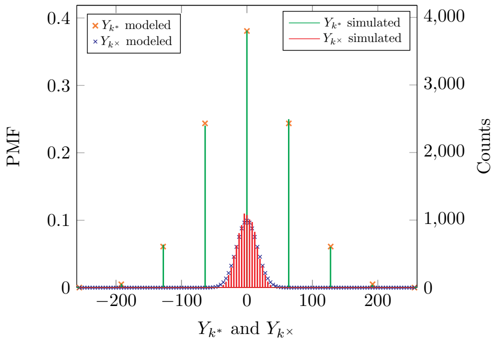

Figure 3: The histogram for the simulation (using 10 thousand trials) matches the theoretical analysis when $(n, m) = (8, 4)$.

DCA success probability. For DCA to succeed, the encoding ε of the target variable must be such that the max absolute correlation over the coordinates $j \in [m]$ for the right key guess k^* is greater than the max absolute correlation over the coordinates $j \in [m]$ and the wrong key guesses $k^\times \in \mathcal{K} \setminus \{k^*\}$. We denote such an event $\mathsf{Succ}_\varepsilon$ in the following, that is

$$\mathsf{Succ}_\varepsilon \;\;:\;\; \max_{j} \,|\mathrm{Cor}(\varphi_i^*, \varepsilon_j \circ \varphi^*)| > \max_{j,\, k^\times} |\mathrm{Cor}(\varphi_i^\times, \varepsilon_j \circ \varphi^*)| \;.$$

Note that the probability that the above event occurs only depends on the random generation of ε, which happens during the *compilation* process of the white-box implementation. If this condition is satisfied, then the attack will succeed provided that the number of computation traces is sufficient to get enough accuracy in the correlation estimation. We first analyze the occurrence probability of $\mathsf{Succ}_\varepsilon$ and then address the trace complexity.

Let us first look at the simpler case of a single j, namely the probability to get $|\mathrm{Cor}(\varphi_i^*, \varepsilon_j \circ \varphi^*)| > \max_{k^\times} |\mathrm{Cor}(\varphi_i^\times, \varepsilon_j \circ \varphi^*)|$. In our idealized model, this amounts to get $\big|Y_{k^*}\big| > \max_{k^\times} \big|Y_{k^\times}\big|$ where Y_{k^*} and $Y_{k^\times}$ are as defined above.

Proposition 1. *Under our idealized model, we have*

$$\Pr\big(|Y_{k^*}| > \max_{k^\times} |Y_{k^\times}|\big) \;\;=\;\; 2 \sum_{0 \le z < 2^{m-2}} \Pr_{\widetilde{\mathcal{HG}}(m)}(z) \cdot \big(1 - 2 \cdot \Phi_{\widetilde{\mathcal{HG}}(n)}(2^{n-m} \cdot z)\big)^{|\mathcal{K}|-1} \;.$$

The proof of Proposition 1 is given in Appendix A. We observe that the probability $\Pr\big(|Y_{k^*}| > \max_{k^\times} |Y_{k^\times}|\big)$ only depends on n, m and $|\mathcal{K}|$ in our idealized model. To illustrate Proposition 1, we plot in Figure 2 this probability for several values of n and m, taking $|\mathcal{K}| = 2^n$ (which would basically occur for a target function of the form $\varphi_k(x) = \varphi'(x \oplus k)$). For instance, for $n = 8, m = 4$, we have more than $\frac{1}{2}$ probability to get $|Y_{k^*}|$ greater than $|Y_{k^\times}|$ for the 255 wrong key guesses $k^\times$. This illustrates why DCA works on nibble encoding of the AES S-box. We also see that for $m = 8$, the probability $\Pr\big(|Y_{k^*}| > \max_{k^\times} |Y_{k^\times}|\big)$ increases with n and also exceeds $\frac{1}{2}$ for $n \ge 13$. This suggests that DCA can also work on byte encodings by targeting an intermediate variable depending on *e.g.* 16 plaintext bits.

Figure 3 further plots the distributions of Y_{k^*} and $Y_{k^\times}$ for $n = 8, m = 4$ (as well as some simulations commented below). We observe that the difference of variances makes the two distributions easily distinguishable. We further observe that whenever $Y_{k^*} \ne 0$, we have a very high probability that $|Y_{k^*}| > |Y_{k^\times}|$. This is because the values taken by Y_{k^*} are multiples of 2^{n-m+2} (which equals 64 for $n = 8, m = 4$), therefore $Y_{k^*} \ne 0$ implies $|Y_{k^*}| \ge 2^{n-m+2}$. On the other hand, the standard deviation of $Y_{k^\times}$, which is close to $2^{n/2}$ according to Equation 3 (*i.e.* around 16 for $n = 8$), might be significantly smaller than

2^{n-m+2}. More generally, if for a small constant q_α, n is chosen such that

$$q_\alpha \cdot \sigma(Y_{k^\times}) \approx q_\alpha \cdot 2^{\frac{n}{2}} \leq 2^{n-m+2} \quad \Leftrightarrow \quad n \geq 2m + 2(\log_2 q_\alpha - 2)$$

we have an overwhelming probability α that $|Y_{k^*}| > |Y_{k^\times}|$.[3] For instance, taking $n \geq 2m+2$ gives $q_\alpha \geq 8$ which (under a Gaussian approximation of $Y_{k^\times}$) implies $\alpha \geq 1 - 10^{-14}$. Consequently, choosing n slightly greater than $2m$ we get

$$\Pr\big(|Y_{k^*}| > \max_{k^\times} |Y_{k^\times}|\big) \approx 1 - \Pr(Y_{k^*} = 0) = 1 - \Pr_{\widetilde{\mathcal{HG}}(m)}(2^{m-2}) \tag{4}$$

where $\Pr_{\widetilde{\mathcal{HG}}(m)}(2^{m-2}) = \binom{2^{m-1}}{2^{m-2}}^2 / \binom{2^m}{2^{m-1}}$.

It can be checked from Figure 2 that $\Pr\big(|Y_{k^*}| > \max_{k^\times} |Y_{k^\times}|\big)$ indeed converges towards the above approximation as n grows and that the convergence is indeed achieved for $n \geq 2m + 2$.

Let us now extend Proposition 1 to the general case of $\mathsf{Succ}_\varepsilon$ where the max is taken over all the coordinates $j \in [m]$. We shall extend our idealized model by assuming that the coordinate functions φ_j are mutually independent random functions of $\mathcal{B}(n)$.

Proposition 2. *Under our idealized model, we have*

$$\Pr\big(\mathsf{Succ}_\varepsilon\big) = \sum_{0 \leq z < 2^{m-2}} \mu(z) \cdot \Big(1 - 2 \cdot \Phi_{\widetilde{\mathcal{HG}}(n)}\big(2^{n-m} \cdot z\big)\Big)^{m\cdot(|\mathcal{K}|-1)}$$

where

$$\mu(z) = \sum_{\ell=1}^{m} \binom{m}{\ell} \cdot \Big(2\Pr_{\widetilde{\mathcal{HG}}(m)}(z)\Big)^{\ell} \cdot \Big(1 - 2\Phi_{\widetilde{\mathcal{HG}}(m)}(z)\Big)^{m-\ell} .$$

The proof of Proposition 2 is given in Appendix B. In Section 4.3, we provide a comparison of $\Pr\big(\mathsf{Succ}_\varepsilon\big)$ in the ideal setting and in a real setting.

As above, taking $n \geq 2m + 2$, we get $|Y_{k^\times}| < 2^{n-m+2}$ with overwhelming probability (for all the wrong key guesses $k^\times$ and coordinates j), and hence $\mathsf{Succ}_\varepsilon$ occurs whenever $Y_{k^*} = \mathrm{B}(\varepsilon_j \circ \varphi^* + \varphi^*_i)$ is non-zero for a single $j \in [m]$. That is

$$\Pr\big(\mathsf{Succ}_\varepsilon\big) \approx 1 - \Pr(Y_{k^*} = 0)^m = 1 - \Pr_{\widetilde{\mathcal{HG}}(m)}(2^{m-2})^m . \tag{5}$$

This approximation is also empirically validated in Section 4.3.

We have analyzed the probability that an internal encoding ε makes it possible for a DCA to succeed. Let us now extend the analysis by considering the full computation trace. Under our idealized model, the latter is composed of the m coordinates of $\varepsilon \circ \varphi^*(X)$ and of $T - m$ samples generated from fresh random functions of $\mathcal{B}(n)$. We have the following corollary of Proposition 2.

Corollary 1. *Let* $\mathsf{Full\text{-}Succ}_\varepsilon$ *denote the event*

$$\mathsf{Full\text{-}Succ}_\varepsilon \;:\; \max_{j \in [T]} |\mathrm{Cor}(\varphi^*_i, V_j)| > \max_{j \in [T],\, k^\times} |\mathrm{Cor}(\varphi^\times_i, V_j)| .$$

Under our idealized model, we have

$$\Pr\big(\mathsf{Full\text{-}Succ}_\varepsilon\big) \geq \sum_{0 \leq z < 2^{m-2}} \mu(z) \cdot \Big(1 - 2 \cdot \Phi_{\widetilde{\mathcal{HG}}(n)}\big(2^{n-m} \cdot z\big)\Big)^{T\cdot(|\mathcal{K}|-1)} \tag{6}$$

where $\mu(z)$ is defined as in Proposition 2.

[3] Here q_α is the quantile of α meaning that we have probability α that $Y_{k^\times}$ is smaller (in absolute value) than q_α times its standard deviation. In particular, α quickly converges towards 1 as q_α grows.

The inequality in Equation 6 directly results from

$$\max_{j\in[T]} |\mathrm{Cor}(\varphi_i^*, V_j)| \geq \max_{j\in[m]} |\mathrm{Cor}(\varphi_i^*, \varepsilon_j \circ \varphi^*)| \ .$$

The rest of the proof is similar to the proof of Proposition 2.

In the above propositions (and corollary), we have exhibited the probability that the exact correlation score is greater for the right key guess k^* than for any wrong key guess $k^\times$. Although this is a necessary condition for DCA to succeed, one further needs to get some estimations of the correlation scores which are accurate enough to ensure the superiority of the right correlation peak. We analyze hereafter the number of traces necessary to meet such a practical condition.

Trace complexity. Let us recall that $\mathrm{Cor}(\varphi_i^*, \varepsilon_j \circ \varphi^*) = 2^{-n} \cdot Y_{k^*}$. We have seen that, with high probability (see Equation 5), we have $Y_{k^*} \neq 0$, and hence $|Y_{k^*}| \geq 2^{n-m+2}$, for at least one coordinate j. We consider hereafter that this event indeed occurs from which we get

$$\delta_{k^*}^{\mathrm{dca}} = \max_{j\in[m]} |\mathrm{Cor}(\varphi_i^*, \varepsilon_j \circ \varphi^*)| \geq 2^{-m+2} \ .$$

Moreover, taking $n \geq 2m+2$, we have seen that the variables $Y_{k^\times}$ are an order of magnitude lower than the 2^{n-m+2}, hence the correlation scores $\delta_{k^\times}^{\mathrm{dca}}$ are an order of magnitude lower than $\delta_{k^*}^{\mathrm{dca}}$. In such a case, the number of traces for a successful correlation attack can be approximated by

$$N \approx cst \cdot \left(\frac{1}{\ln\left(\frac{1+\rho}{1-\rho}\right)} \right)^2 \approx \ cst' \cdot \left(\frac{1}{\rho}\right)^2 \ ,$$

where cst and cst' are small constants (depending on the desired success probability) and where ρ is the correlation of the right key guess, *i.e.* $\rho = \delta_{k^*}^{\mathrm{dca}}$ in our context. The first approximation is due to Mangard [Man04] and the second is a Taylor approximation which is sound as long as ρ is small enough (which holds in our case for typical values of m). We hence get a trace complexity of $N = O(2^{2m})$.

4.3 Simulations

In order to verify that our ideal analysis soundly captures the behavior of an actual DCA, we perform several attack simulations taking an AES S-box output as target variable, with $n = 8$, and for different encoding size $m = \{4, 5, 6, 7, 8\}$. Specifically, we look at the distributions of $Y_{k^*} = \mathrm{B}(\varepsilon_j \circ \varphi + \varphi_i^*)$ and $Y_{k^\times} = \mathrm{B}(\varepsilon_j \circ \varphi + \varphi_i^\times)$, and the probabilities of events: $\big|Y_{k^*}\big| > \max_{k^\times} \big|Y_{k^\times}\big|$ and $\mathsf{Succ}_\varepsilon$, with $\varphi_k(x) = S^{(m)}(x \oplus k)$ where $S^{(m)}$ is the AES S-box shrunk to its m least significant bits (and hence φ_k is a $(8, m)$-VBF). For all settings, our simulation results are averaged over 10,000 trials and ε is a fresh random m-bit bijection in each trial. The full simulations are done according to the procedures depicted in Figure 9 in Appendix C.

As an illustration, we plot the histogram for $(n, m) = (8, 4)$ in Figure 3, which demonstrates that our theoretical analysis on distributions of Y_{k^*} and $Y_{k^\times}$ matches the real distributions obtained in a DCA experiment. We further compare in Table 1 the ideal and simulation settings for the probabilities $\Pr\big(|Y_{k^*}| > \max_{k^\times} |Y_{k^\times}|\big)$ and $\Pr\big(\mathsf{Succ}_\varepsilon\big)$. We can observe that the figures obtained through our ideal analysis match pretty well the simulation results. For instance, when $(n, m) = (8, 4)$, the probabilities of $|Y_{k^*}| > \max_{k^\times} |Y_{k^\times}|$ are about 0.6071 (ideal setting) and 0.6194 (simulation). Note that the difference is of same order of magnitude (*i.e.* 10^{-2}) as the precision of the simulation results (based on $10,000$ trials).

Table 1: The simulation and theoretical (ideal) analysis results for $n = 8, m = \{4, 5, 6, 7, 8\}$ by using AES-128 first round S-box as the selection function.

(n, m)	$\Pr(\lvert Y_{k^*}\rvert > \max_{k^\times} \lvert Y_{k^\times}\rvert)$		$\Pr(\mathsf{Succ}_\varepsilon)$	
	ideal	**simulation**	**ideal**	**simulation**
(8,4)	0.6071	0.6194	0.9264	0.9722
(8,5)	0.2837	0.2859	0.7598	0.8032
(8,6)	0.1259	0.1281	0.3556	0.3749
(8,7)	0.0305	0.0299	0.0716	0.0723
(8,8)	0.0027	0.0021	0.0025	0.0020

We also verify the soundness of the approximation in Equation 4 and Equation 5 when one takes $n = 2m + 2$. For this purpose, we compare in Table 2 the probabilities of the events $\lvert Y_{k^*}\rvert > \max_{k^\times} \lvert Y_{k^\times}\rvert$ and $\mathsf{Succ}_\varepsilon$ obtained from our approximations, from our propositions in the ideal model, and from simulations. The simulations are based on 10,000 attack trials, where φ_k is defined as $\varphi_k(x) = \varphi(x \oplus k)$ for some (n, m)-VBF φ randomly picked in each trial.

Table 2: The simulation and theoretical (ideal) analysis results for $n = 2m + 2$ where $m = \{3, 4, 5, 6, 7\}$ by targeting at a n-bit random bijection.

(n, m)	$\Pr(\lvert Y_{k^*}\rvert > \max_{k^\times} \lvert Y_{k^\times}\rvert)$			$\Pr(\mathsf{Succ}_\varepsilon)$		
	Equation 4	**ideal**	**simulation**	Equation 5	**ideal**	**simulation**
(8,3)	0.4857	0.4857	0.4853	0.8640	0.8640	0.8828
(10,4)	0.6193	0.6193	0.6123	0.9790	0.9790	0.9736
(12,5)	0.7244	0.7244	0.7141	0.9984	0.9984	0.9960
(14,6)	0.8029	0.8029	0.8027	0.999941	0.999941	1.0000
(16,7)	0.8598	0.8598	0.8615	0.999998934	0.999998934	1.0000

4.4 Practical Attack Experiments

4.4.1 Target Implementations

The NSC Variant. In NoSuchCon (NSC) 2013, a Windows binary embedding with a white-box implementation of AES-128 protected by external and internal encodings, was published by Vanderbéken as a challenge.[4] Due to the presence of external encodings, the authors of [BHMT16] failed to to break this challenge with DCA. Besides the challenge binary, Vanderbéken also published its generator, which led to the publication of multiple variants.[5] One interesting variant, referred as *the NSC variant* in the following, is an implementation protected with internal encodings only (*i.e.* an implementation of the standard AES-128). However, since it makes use of byte encodings, this implementation was believed to resist DCA.[5]

Lee's CASE 1 Implementation. Recently, Lee *et al.* published a white-box implementation of AES against DCA attack [LKK18] which consists in applying additional countermeasures to the original Chow *et al.*'s implementation. Three protection cases are suggested: CASE 1 uses some masking techniques before applying internal encodings to protect the first and last round of the implementation; CASE 2 and CASE 3 work

[4] See http://www.nosuchcon.org/2013/ and http://seclists.org/fulldisclosure/2013/Apr/133.

[5] See https://github.com/SideChannelMarvels/Deadpool/tree/master/wbs_aes_nsc2013_variants.

as CASE 1 but with larger internal encodings (byte instead of nibbles) applied to more variables in the outer rounds. An implementation (under the form of a binary program) was made publicly available for CASE 1[6] but not for CASE 2 nor CASE 3. In this work, we do not intend to give a full cryptanalysis of Lee's proposal, but we point out that the masking technique can be bypassed by targeting an input byte of the second round, *i.e.* an output byte of the first-round MixColumn, which allows us to apply a simple DCA attack as described below.

4.4.2 Target Variables

In all our experiments, we select one MixColumn output byte in the first round as our target. Such a byte s satisfies

$$\begin{aligned} s &= \mathrm{MixColumn}\big(S(x_1 \oplus k_1^*), S(x_2 \oplus k_2^*), S(x_3 \oplus k_3^*), S(x_4 \oplus k_4^*)\big) \\ &= S(x_1 \oplus k_1^*) \oplus S(x_2 \oplus k_2^*) \oplus 2 \cdot S(x_3 \oplus k_3^*) \oplus 3 \cdot S(x_4 \oplus k_4^*) \ , \end{aligned}$$

where S denotes the AES S-box, (x_1, x_2, x_3, x_4) are four plaintext bytes, and the above multiplications (by 2 and 3) are on the field $\mathbb{F}_{256}$.

In order to reduce the key space for guessing such a byte, we select some random plaintext with fixed value for x_3 and x_4 (specifically $x_3 = x_4 = 0$). Doing so, the byte s can be rewritten as $s = S(x_1 \oplus k_1^*) \oplus S(x_2 \oplus k_2^*) \oplus c^*$ for some (secret) constant c^* and the encoded byte $\varepsilon(s)$ is identically distributed (over a random choice of ε) to the byte $\varepsilon\big(S(x_1 \oplus k_1^*) \oplus S(x_2 \oplus k_2^*)\big)$. We then make a 2-byte key guess (k_1, k_2) and calculate the predictions of the target byte as

$$\varphi_{k_1,k_2}(x_1, x_2) = S(x_1 \oplus k_1) \oplus S(x_2 \oplus k_2)$$

for each plaintext with bytes (x_1, x_2). Our selection function φ is hence a $(16, 8)$-VBF and the size of the key space is 2^{16}.

Although we only focus on recovering two key bytes, we could easily repeat this attack by swapping the fixed pair of bytes to recover k_3^* and k_4^* and do the same for the 3 other MixColumn computations. Moreover, we can fix a pair of bytes in each column while collecting the traces, so that only two sets of traces would be sufficient to recover the full key.

4.4.3 Attack Results

For each implementation, some preliminary analysis of the binary allowed us to obtain the addresses of the executable code segment in the virtual memory. We were then able to collect the bytes written on the stack during an execution. For such a purpose we used the SideChannelMarvels Tracer tool.[7] We thus obtained computation traces composed of 1850 and 21536 byte samples for the NSC variant and Lee's CASE 1 implementation respectively. Each of these samples was split into bits to get binary traces from which we removed the duplicated columns. We finally obtained binary computation traces composed of 6077 and 24012 samples for the NSC variant and Lee's CASE 1 implementation respectively.

For the NSC variant, we collected 1,800 traces from which we computed the correlation traces for each key guess as described in Section 4.1. As an illustration, Figure 4 plots the correlation traces obtained when the least significant bit of the target variable is used as prediction function. The attack was conducted for the 8 prediction bits, and the correct key guess was ranked first (among the 2^{16} guesses) in 4 out of 8 attacks.

One can observe multiple peaks in the correlation trace for the right key guess that might correspond to different manipulations of the target variable through different encodings

[6] See https://github.com/SideChannelMarvels/Deadpool/tree/master/wbs_aes_lee_case1.

[7] See https://github.com/SideChannelMarvels/Tracer.

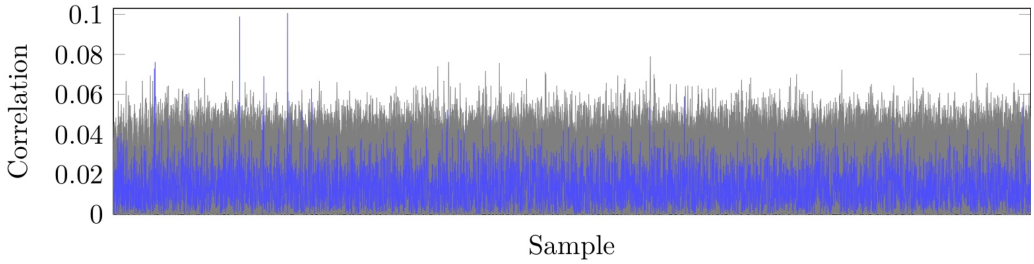

Figure 4: DCA correlation traces on the NSC variant for the good key guess (in blue) and for 256 (out of $2^{16}-1$) incorrect key guesses (in gray).

ε. It is worth noting that, as exhibited in our analysis, these peaks converge towards correlation scores which are multiples of 2^{-m+2} *i.e.* of $\frac{1}{64}$ for $m=8$. The first clearly distinguishable peak is around $\frac{5}{64}\approx 0.078$ while the two next peaks are around $\frac{6}{64}\approx 0.094$ (the match is not perfect due to the estimation error).

Using 1800 traces implies that the average noise in the correlation trace is around $\sqrt{1/1800}\approx 0.02$ (which reaches 0.06 when we take the max over 256 correlation traces as we observe on Figure 4). That is why only the peaks around $\frac{6}{64}$ are clearly distinguishable from the noise and that is why the maximal peak is reached for the good key guess for only 4 selection bits out of 8. Taking more traces would certainly ensure that smaller multiples of $\frac{1}{64}$ could also be distinguishable from the noise which would increase the number of prediction bits (up to 8) for which the attack works.

For Lee's CASE 1 implementation, we were able to mount a successful attack using 4000 traces. The obtained correlation traces are very similar to Figure 4 which all includes a few distinguishable peaks for the right key guess.

4.5 Discussion

4.5.1 The Case of Linear Encodings

We address hereafter the case of *linear encodings*, which are encodings ε that can be expressed as

$$\varepsilon(s) = \boldsymbol{A}\cdot(s_1,\cdots,s_m)^{\mathsf{T}} \oplus \boldsymbol{b}^{\mathsf{T}} \ .$$

where $(s_1,\cdots,s_m)$ is the binary representation of s, $\boldsymbol{A}=(a_{j,\ell})_{j,\ell\in[m]}\in\mathbb{F}_2^{m\times m}$ is an invertible binary matrix and $\boldsymbol{b}=(b_j)_{j\in[m]}\in\mathbb{F}_2^m$ is a binary vector. For such a linear encoding, we get $\varepsilon_j\circ\varphi^*(x)=b_j+\sum_{\ell=1}^m a_{j,\ell}\,\varphi_\ell^*(x)$ for every $j\in[m]$ and hence

$$\mathrm{Cor}(\varphi_i,\varepsilon_j\circ\varphi^*)=\frac{1}{2^n}\cdot B\Big(\varphi_i+b_j+\sum_{\ell=1}^m a_{j,\ell}\,\varphi_\ell^*\Big)\ .$$

Then we differentiate three cases:

- if $\varphi=\varphi^*$ (*i.e.* the key guess is correct) and if $a_{j,i}=1$ and $a_{j,\ell}=0$ for every $\ell\in[m]\setminus\{j\}$, then we have $|\mathrm{Cor}(\varphi_i,\varepsilon_j\circ\varphi^*)|=\frac{1}{2^n}\cdot|B(b_j)|=1$;
- if $\varphi=\varphi^*$ (*i.e.* the key guess is correct) and if $a_{j,\ell}=1$ for some $\ell\in[m]\setminus\{j\}$, then $\varphi_i^*+b_j+\sum_{\ell=1}^m a_{j,\ell}\,\varphi_\ell^*$ is balanced implying $|\mathrm{Cor}(\varphi_i,\varepsilon_j\circ\varphi^*)|=0$;
- if $\varphi=\varphi^\times$ (*i.e.* the key guess is incorrect), under our idealized model, we have $|\mathrm{Cor}(\varphi_i,\varepsilon_j\circ\varphi^*)|=\frac{1}{2^n}|B(\varphi_i+f)|$ where $f=b_j+\sum_{\ell=1}^m a_{j,\ell}\,\varphi_\ell^*$ is a random function of $\mathcal{B}(n)$ (since f is a linear combination of the coordinates of a random balanced VBF

φ^*) and independent of φ_i, namely $|\mathrm{Cor}(\varphi_i, \varepsilon_j \circ \varphi^*)|$ is distributed as the variable $\frac{1}{2^n} \cdot Y_{k^\times}$ (see Equation 2).

We can deduce that if the matrix $\boldsymbol{A}$ has at least one row –say the jth row– of Hamming weight 1, *i.e.* with a single coefficient to $a_{j,i} = 1$, then for the corresponding $i \in [m]$, we have $|\mathrm{Cor}(\varphi_i, \varepsilon_j \circ \varphi^*)| = 1$ which implies that DCA (targeting the ith bit of φ) will succeed with overwhelming probability. If no such row of Hamming weight 1 occurs, then the right guess correlation is indistinguishable from the wrong guess correlations and DCA fails with high probability.

There is a certain probability that a random encoding happens to be a linear encoding. This is especially likely when $m = 2, 3$. In particular, when $m = 2$, there are only 6 possible $\varepsilon_j(s_1, s_2)$ with ANF

$$s_1 + b \ , \ s_2 + b \ , \ \text{and } s_1 + s_2 + b \ , \ \text{where } b \in \mathbb{F}_2 \ .$$

Hence, given i and j, all the possible encodings are linear, and only $s_i + b$ satisfies the condition $a_{j,i} = 1$ and $a_{j,\ell} = 0$ for every $\ell \in [m] \setminus \{j\}$. This high probability of getting a linear encoding implies that the success of DCA against encodings of size $m = 2, 3$ is less likely than for greater value of m as indicated by Proposition 2..

4.5.2 Comparison to Previous Analysis

In [BBMT18, BBB+17], Bock *et al.* conduct an analysis to explain the ineffectiveness of linear and/or nibble encodings against DCA. In comparison, our analysis covers random (non-linear) encodings of any size m. Regarding the cases of linear encodings and (non-linear) nibble encodings, our analysis is consistent with the results of Bock *et al.* while providing more formal statements (under some ideal assumption) and close formulas for the success rate of DCA with respect to the attack parameters $(n, m, |\mathcal{K}|)$.

More precisely, for the case of linear encodings, our analysis of Section 4.5.1 is similar to Theorem 1 in [BBMT18, BBB+17], but the latter does not deal with the correlation scores of wrong key guesses, whereas our analysis characterizes these scores under an ideal assumption. For the case of nibble encodings, our analysis exhibits the distribution of the right guess correlation score as $\mathrm{Cor}(\varepsilon_j \circ \varphi^*, \varphi_i^*) = \frac{1}{4} \cdot N_{00}^{\varepsilon_j \circ \varphi^*, \varphi_i^*} - 1$, with $N_{00}^{\varepsilon_j \circ \varphi^*, \varphi_i^*} \sim \widetilde{HG}(4)$. This result implies in particular that the possible correlation scores for the right guess are multiples of $\frac{1}{4}$, which is the purpose of Theorem 2 in [BBMT18]. Besides the correlation scores for the good key guess, our analysis further characterizes the distribution for wrong guess correlation scores, whereas this distribution is considered “close to 0” in [BBMT18].

Bock *et al.* also look at the empirical distribution of the correlation scores for the correct key guess with 10,000 attack simulations on the AES S-box protected by nibble encodings. In the considered scenario, the max correlation score is taken over the 8 predicted bits and the 4 output bits of the encoding. Tweaking Proposition 2 to this case, the probability to have a correlation peak of $\frac{1}{4}(4 - z)$ is given by

$$\mu(z) = \sum_{\ell=1}^{32} \binom{32}{\ell} \cdot \Big(2\mathrm{Pr}_{\widetilde{\mathcal{HG}}(4)}(z)\Big)^{\ell} \cdot \Big(1 - 2\Phi_{\widetilde{\mathcal{HG}}(4)}(z)\Big)^{32-\ell} \ .$$

Table 3 compares the above formula to the figures given in [BBMT18] which shows a good match between our formal analysis and the simulation results given by Bock *et al.*.

5 Collision Attack

Generating and analyzing collisions in a computation is a common attack technique in the side-channel context [SLFP04, MME10]. In this section, we propose a new class of gray-box DCA-like collision attacks to break white-box implementations protected by internal

Table 3: Simulation results in [BBMT18] *vs.* our formula for nibble encodings.

Score	Count [BBMT18]	Probability
1	55	$\mu(0) = 0.0050$
0.75	2804	$\mu(1) = 0.2724$
0.50	7107	$\mu(2) = 0.7118$
0.25	34	$\mu(3) = 0.0108$
0	0	$\mu(4) = 0.0000$

encodings. We first give in Section 5.1 a formal description of our collision distinguisher within the previously introduced passive attack model. Then we show in Section 5.2 how it can effectively break the NSC variant and Lee's CASE 1 white-box implementations. We finally give a theoretical analysis of the success probability and trace complexity of our collision attack in Section 5.3.

5.1 Collision Attack Distinguisher

Following the passive attack model introduced in Section 3.2, the adversary first collects N computation traces $\left(\boldsymbol{v}^{(i)}\right)_i$ corresponding to N inputs $\left(x^{(i)}\right)_i$ for the target function φ. Then for each pair of inputs $\left(x^{(i_1)}, x^{(i_2)}\right)$ where $i_1, i_2 \in [N], i_1 \neq i_2$, and their corresponding computation traces $\left(\boldsymbol{v}^{(i_1)}, \boldsymbol{v}^{(i_2)}\right)$, the adversary computes a *collision computation trace* (CCT):

$$\boldsymbol{w}^{(i_1,i_2)} = \left(w_1^{(i_1,i_2)}, w_2^{(i_1,i_2)}, \cdots, w_T^{(i_1,i_2)}\right),$$

with $w_j^{(i_1,i_2)} = v_j^{(i_1)} \odot v_j^{(i_2)}$ for every $j \in [T]$ where the operator $\odot$ is defined as

$$a \odot b := \begin{cases} 1 & \text{if } a = b, \\ 0 & \text{otherwise.} \end{cases}$$

Namely, the collision computation trace for indexes (i_1, i_2) has a 1 at the jth sample position iff a collision occurs between the jth samples of the computation traces $\boldsymbol{v}^{(i_1)}$ and $\boldsymbol{v}^{(i_2)}$. Similarly, the *collision prediction* for a key guess $k \in \mathcal{K}$ and input values $\left(x^{(i_1)}, x^{(i_2)}\right)$ is defined as

$$\psi_k\left(x^{(i_1)}, x^{(i_2)}\right) := \varphi_k\left(x^{(i_1)}\right) \odot \varphi_k\left(x^{(i_2)}\right).$$

The collision distinguisher for a key guess k is then defined as the maximal correlation between the CCT and the corresponding collision prediction for k, *i.e.*,

$$\delta_k^{\mathrm{ca}} = \max_{j \in [T]} \operatorname{Cor}\Big(\psi_k\left(X^{(i_1)}, X^{(i_2)}\right), W_j^{(i_1,i_2)}\Big) \ .$$

As for DCA, the above correlation coefficient is estimated based on the collected samples $x^{(i)}$ and $\boldsymbol{w}^{(i,j)}$, for $i, j \in [N]$.

The soundness of our collision attack against internal encodings can be summarized with the following observation: if some sensitive variable collides for a pair of inputs, so does the corresponding encoded variable in the computation trace. Conversely, if some sensitive variable does not collide for a pair of inputs, neither does the corresponding encoded variable in the computation trace. As a consequence, there is a perfect match between the collision prediction and the target sample in the CCT for the correct key guess (implying a correlation score to 1) whereas this should not hold for an incorrect key guess.

For a typical selection function φ, the instances $(\varphi_k)_k$ corresponding to the different key guesses behave like independent random functions. Hence, the success probability of our collision attack quickly grows with the number of collision pairs, as we analyze in more details in Section 5.3.

5.2 Practical Attack Experiments

To validate our intuition, we first experiment our collision attack against the NSC variant and Lee's CASE 1 white-box implementations described in Section 4.4.1. We use the same target variable as in our DCA experiments, which is a MixColumn output in the first round, turned into a $(16, 8)$-VBF –with a key space of size 2^{16}– by fixing two input bytes (see Section 4.4.2 for details).

Our collision attack recovers the two key bytes using 60 computation traces only (which is to be compared with the 1800 traces required by DCA). As an illustration, Figure 5 plots the correlation traces for the correct key guess and for 256 (out of $2^{16} - 1$) incorrect key guesses. We observe some correlation peaks to 1 for the correct key. For the key guess ranked second, we observe a few peaks reaching 0.5 whereas for the other key guesses most of the peaks are around or lower than 0.25.

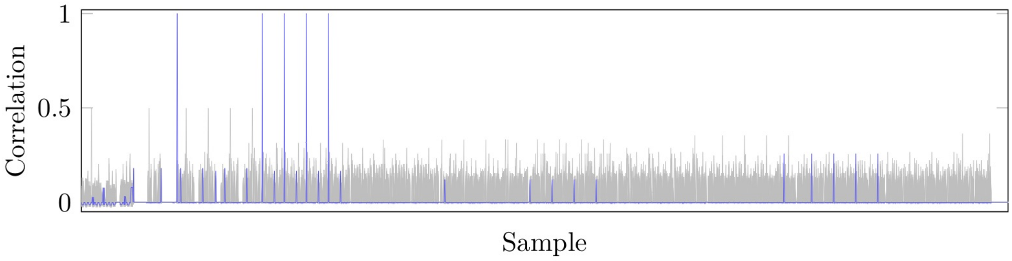

Figure 5: Collision attack traces on the NSC variant for the good key guess (in blue) and for 256 (out of $2^{16} - 1$) incorrect key (in gray).

The same collision attack has been applied to Lee's CASE 1 implementation and could also recover the correct (two-byte) key guess using 60 traces. Although no implementations of Lee's CASE 2 & 3 are publicly available, we note that these variants (which consist in applying byte encodings to internal rounds) should not prevent our collision attack.

5.3 Theoretical Analysis

We analyze hereafter the success and trace complexity of our collision attack in the idealized model. For this purpose, we first introduce a random experiment that we shall call the *balls-and-containers game.*

The balls-and-containers game. In an (α, β, γ)-balls-and-containers game experiment, a player randomly places α different balls in γ different containers of β slots each, such that, at each step, the random placement of a ball is done uniformly among the remaining free slots.[8] As an illustration, the outcome of 4 independent experiments of the $(5, 3, 6)$-balls-and-containers game is represented in Figure 6.

We say that a container *collides* when it contains more than one ball at the end of an experiment. For instance, the 3rd and 5th containers collide in the first experiment in Figure 6 whereas the other containers do not collide. We further say that the outcomes of two experiments are *isomorphic* whenever a reordering of the containers in one outcomes yields a distribution of the balls among the containers which is the same as for the other outcome. For instance, the outcomes of the two first experiments in Figure 6 are isomorphic.

Collision probability. Consider a (α, β, γ)-balls-and-containers game for which $\gamma > \alpha$ (*i.e.* there are more containers than balls). Let Col be the event that at least one container collides. We have

[8] In particular a container with one or several ball(s) has a lower probability to receive a new ball than a container with only free slots.

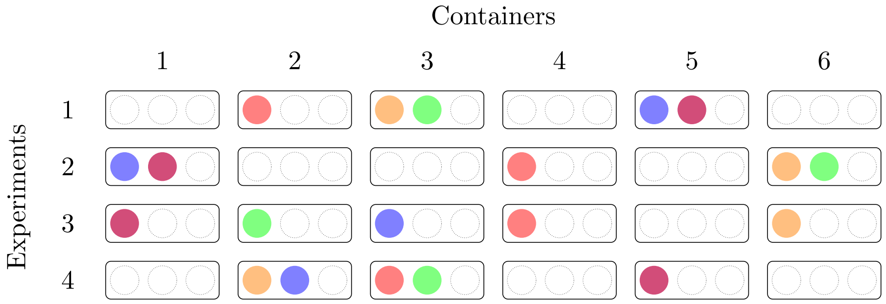

Figure 6: Outcome of 4 independent experiments of the $(5,3,6)$-balls-and-containers game, in which different balls are in different colors.

$$\Pr(\neg \mathsf{Col}) = 1 - \Pr(\mathsf{Col}) = g(\alpha,\beta,\gamma) \ ,$$

where

$$g(\alpha,\beta,\gamma) := \prod_{i=1}^{\alpha-1} \frac{\beta(\gamma-i)}{\gamma\beta-i} \ . \tag{7}$$

Lemma 2. *If $\alpha < \beta$ and $\alpha < \gamma$, we have*

$$g(\alpha,\beta,\gamma) < \exp\left(-\frac{(\alpha-2)(\alpha-1)}{2\gamma}\right) \ .$$

The proof is provided in Appendix D.

Isomorphism probability. Let us denote Iso the event that two independent experiments of the (α,β,γ)-balls-and-containers game are isomorphic. Given γ' the number of containers with at least one ball in the first experiment, we have

$$\Pr(\mathsf{Iso}) \leq \prod_{i=1}^{\gamma'-1} \frac{\beta(\gamma-i)}{\gamma\beta-i} \cdot \prod_{i=\gamma'}^{\alpha-1} \frac{\beta-1}{\gamma\beta-i} \leq g(\alpha,\beta,\gamma) \ .$$

The above probability can be interpreted as follows. In order to have the second experiment isomorphic to the first one, the two following shall occur:

(1) taking one ball from each γ' non-empty container in the first experiment, one must get that these γ' balls are placed in different containers in the second experiment;

(2) each of the remaining balls must end in a specific container (with at most $\beta - 1$ free slots) to satisfy the isomorphic property.

The first inequality comes from the fact that there might be less than $\beta - 1$ remaining free slots in a container for the placing of the remaining balls. The second inequality holds by definition of g (see Equation 7) and from $\beta(\gamma - i) > \beta - 1$ (since $\gamma > \alpha$).

Collision attack success probability. We analyze our collision attack under an idealized model as the one considered for our analysis of DCA (see Section 4.2). In particular, the functions $(\varphi_k)_{k\in\mathcal{K}}$ are assumed to be mutually independent random balanced (n,m)-VBFs. Unlike DCA, the collision attack does not split the samples in the computation trace into bits. We hence consider that the definition space of the V_j's matches the encoding definition space, *i.e.* $\mathcal{V} = \mathbb{F}_2^m$. For some $j^* \in [T]$, we have $V_{j^*} = \varepsilon \circ \varphi^*(X)$, for the other

$j \in [T] \setminus \{j^*\}$, we ideally assume that the samples can be expressed as $V_j = f_j(X)$ where the f_j's are uniform random balanced (n, m)-VBFs.

We first consider the success event

$$\mathsf{Succ} \ : \ \mathrm{Cor}\Big(\psi_{k^*}\big(X^{(i_1)}, X^{(i_2)}\big), W_{j^*}^{(i_1,i_2)}\Big) > \max_{k^\times} \mathrm{Cor}\Big(\psi_{k^\times}\big(X^{(i_1)}, X^{(i_2)}\big), W_{j^*}^{(i_1,i_2)}\Big) \ ,$$

i.e. the correlation is maximal for the correct key guess at the right sample index j^*. Note that for j^*, we have

$$W_{j^*}^{(i_1,i_2)} = \varepsilon \circ \varphi^*(X^{(i_1)}) \odot \varepsilon \circ \varphi^*(X^{(i_2)}) = \varphi^*(X^{(i_1)}) \odot \varphi^*(X^{(i_2)}) = \psi_{k^*}\big(X^{(i_1)}, X^{(i_2)}\big) \ .$$

For some given set of inputs $(x^{(i)})_{i \in N}$, the above success event relies on two events E_1 and E_2, with $\mathsf{Succ} = E_1 \cap E_2$, which are defined as

$$E_1 \ : \ \exists\, (i,j),\ 1 \leq i < j \leq N,\ \text{s.t. } \varphi^*(x^{(i)}) = \varphi^*(x^{(j)}) \ ,$$

and

$$E_2 \ : \ \forall\, k^\times \in \mathcal{K} \setminus \{k^*\},\ \exists\, (i,j),\ 1 \leq i < j \leq N,\ \text{s.t. } \psi_{k^*}(x^{(i)}, x^{(j)}) \neq \psi_{k^\times}(x^{(i)}, x^{(j)}) \ .$$

The event E_1 ensures that the collision predictions $\psi_{k^*}\big(x^{(i)}, x^{(j)}\big)$ are not all equal to zero for the right key guess, which must hold so that the correlation score for k^* is well defined.[9] The event E_2 ensures that for all the wrong key guesses, $(\psi_{k^\times}(x^{(i)}, x^{(j)}))_{i,j}$ does not perfectly match $(\psi_{k^*}(x^{(i)}, x^{(j)}))_{i,j}$, which implies a correlation score strictly lower than 1.

Assuming that we have $N < 2^m$, we can express the collision attack success in terms of balls-and-containers game experiments, by considering

- the inputs $(x^{(i)})_{i \in N}$ as N different balls,
- the output values of φ as 2^m different containers,
- the number of preimages of a given output through φ as the 2^{n-m} slots in each container.

Then each key guess k gives rise to a (α, β, γ)-balls-and-containers game experiment with $\alpha = N$, $\beta = 2^{n-m}$, $\gamma = 2^m$ where the randomness of φ_k acts as a random placement of the inputs $(x_i)_{i \in N}$ in the 2^m output values with a maximum of 2^{n-m} slots per output value (which results from the balanceness of φ_k). The mutual independence of the $(\varphi_k)_{k \in \mathcal{K}}$ implies the mutual independence of the balls-and-containers game experiments.

The event E_1 then holds if at least one container collides in the experiment corresponding to k^*, *i.e.*,

$$\Pr(E_1) = \Pr(\mathsf{Col}) = 1 - g(N, 2^{n-m}, 2^m) \ . \tag{8}$$

On the other hand, the event E_2 holds if none of the experiments for $k^\times \in \mathcal{K} \setminus \{k^*\}$ is isomorphic to the experiment for k^*. The mutual independence of these experiments imply

$$\Pr(E_2) = \big(1 - \Pr(\mathsf{Iso})\big)^{|\mathcal{K}|-1} \geq \big(1 - g(N, 2^{n-m}, 2^m)\big)^{|\mathcal{K}|-1} \ . \tag{9}$$

Proposition 3. *Under our idealized model, we have*

$$\Pr(\mathsf{Succ}) \geq \big(1 - g(N, 2^{n-m}, 2^m)\big)^{|\mathcal{K}|} \geq 1 - |\mathcal{K}| \cdot \exp\left(-\frac{(N-2)(N-1)}{2^{m+1}}\right) \ .$$

[9] In principle we should also ensure that the collision predictions for the right key guess are not all equal to one, *i.e.* the inputs $(x^{(i)})_{i \in N}$ do not all map to the same output through φ_{k^*}, but this shall occur with overwhelming probability so we neglect this requirement.

The proposition is a direct consequence of Equation 8 and Equation 9 (first inequality), and Lemma 2 (second inequality).

Let us now extend the analysis by considering the full computation trace. Under our idealized model, the latter is composed of $\varepsilon \circ \varphi^*(X)$ and of $T-1$ samples generated from fresh random balanced (n,m)-VBFs. We have the following corollary of Proposition 3.

Corollary 2. *Let us denote* $\mathsf{Full\text{-}Succ}$ *the success event*

$$\mathsf{Full\text{-}Succ} \ : \ \max_{j\in[T]} \mathrm{Cor}\Big(\psi_{k^*}\big(X^{(i_1)}, X^{(i_2)}\big), W_j^{(i_1,i_2)}\Big) \\ > \max_{j\in[T],k^\times} \mathrm{Cor}\Big(\psi_{k^\times}\big(X^{(i_1)}, X^{(i_2)}\big), W_j^{(i_1,i_2)}\Big) \ .$$

Under our idealized model, we have

$$\Pr(\mathsf{Full\text{-}Succ}) \geq \big(1 - g(N, 2^{n-m}, 2^m)\big)^{T\cdot|\mathcal{K}|} \geq 1 - T\cdot|\mathcal{K}|\cdot\exp\left(-\frac{(N-2)(N-1)}{2^{m+1}}\right) \ . \quad (10)$$

Corollary 2 is a straightforward extension of Proposition 3 where $(|K|-1)+1$ in the exponent is replaced by $T\cdot(|K|-1)+1$ which directly implies the above inequality.

Trace complexity. From the above analysis, we can easily deduce the trace complexity of our collision attack. Let λ be some parameter such that one wants to achieve a success probability $1-10^{-\lambda}$. By Corollary 2, taking

$$N = \sqrt{2^{m+1}\big(\lambda\log 10 + \log T + \log|\mathcal{K}|\big)} + 1 \ , \quad (11)$$

implies $\Pr(\mathsf{Full\text{-}Succ}) \geq 1-10^{-\lambda}$. Given a (high) success probability, a trace size and a key space, the number of required computation traces is hence $N = \Theta(2^{\frac{m}{2}})$, which is a significant improvement over DCA for which we have $N = O(2^{2m})$.

In order to illustrate our analysis, Figure 7 plots the lower bound on the success probability (Equation 10) for $n=16$, $m=8$, $|\mathcal{K}| = 2^n$, and $T \in \{1, 10^3, 10^6\}$. We see that multiplying the size of the computation trace by a factor 1000 only implies a small gap (less than 20) in the number of required traces. In order to illustrate the tightness of the bounds, we further plot the middle lower bound in Equation 10, *i.e.* $(1-g(N,2^{n-m},2^m))^{T\cdot|\mathcal{K}|}$, as well as the lower bound obtained by a straight application of Lemma 2. We observe that our explicit lower bound only implies a gap of 5 in the number of required computation traces, which is fairly tight.

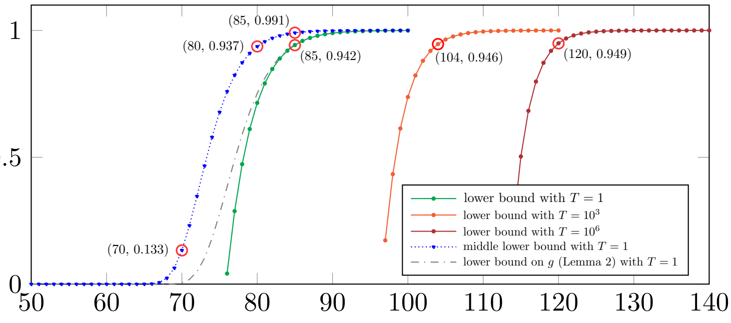

Figure 7: Success probability lower bound (Equation 10) over an increasing N for $n = 16$, $m = 8$, and $|\mathcal{K}| = 2^n$.

6 Mutual Information Analysis

Mutual Information Analysis (MIA) was introduced in the side-channel context for an adversary that has very limited knowledge about the leakage distribution and how it relates to computed data [GBTP08, BGP+11]. In particular, MIA can deal with any kind of –possibly uncommon, odd, or complex– leakage function. It therefore naturally applies in the white-box context to attack implementations protected with internal encodings since the latter can be thought of as particular cases of –especially complex– leakage functions. We first recall the MIA distinguisher in Section 6.1, then we give a brief analysis on its behavior in the considered white-box setting in Section 6.2, and finally we present experimental results in Section 6.3.

6.1 MIA Distinguisher

The MIA distinguisher for a key guess k is calculated as the maximal mutual information between the prediction $\varphi_k(X)$ and each trace sample V_j, that is

$$\delta_k^{\text{mia}} = \max_{j \in [T]} \mathrm{I}\left(\varphi_k(X); V_j\right)$$

The basic notions of information theory are recalled in Appendix E. Note that unlike the side-channel context in which evaluating the mutual information usually involves complex pdf estimation methods, we are only dealing with discrete variables here which makes the practical evaluation simpler.

6.2 Analysis and Improvement

In practice the adversary computes the MIA distinguisher based on sample values. In the following, we shall use denote $\hat{\mathrm{I}}$ and $\hat{\mathrm{H}}$ the sample versions of the mutual information and the entropy which are computed based on a uniform random selection of the inputs $(x^{(i)})_{i \in [N]}$.

Let j^* be the sample index such that $V_{j^*} = \varepsilon \circ \varphi_{k^*}(X)$. For clarity, we abuse notations by skipping the parameter k in the selection function by letting $\varphi = \varphi_k$, $\varphi^* = \varphi_{k^*}$ and $\varphi^\times = \varphi_{k^\times}$. We have

$$\hat{\mathrm{I}}\big(\varphi(X); V_{j^*}\big) = \hat{\mathrm{I}}\big(\varphi; \varepsilon \circ \varphi^*\big) = \hat{\mathrm{I}}\big(\varphi; \varphi^*\big)$$

where we drop the argument X in φ and φ^* for the sake of clarity, and where the last equality holds by the bijectivity of ε. Let us look at the success event that, for the right sample index j^*, the mutual information score is the greatest for the correct key guess, that is

$$\mathsf{Succ} \ : \ \hat{\mathrm{I}}\big(\varphi^*; \varphi^*\big) \geq \max_{k^\times} \hat{\mathrm{I}}\big(\varphi^\times; \varphi^*\big) \ .$$

For the correct key guess, we have

$$\hat{\mathrm{I}}\big(\varphi^*; \varphi^*\big) = \hat{\mathrm{H}}\big(\varphi^*\big) \xrightarrow{N \to \infty} \mathrm{H}\big(\varphi^*\big) = m \ .$$

On the other hand, for an incorrect key guess, we have

$$\hat{\mathrm{I}}\big(\varphi^\times; \varphi^*\big) = \hat{\mathrm{H}}\big(\varphi^*\big) - \hat{\mathrm{H}}\big(\varphi^\times | \varphi^*\big) \ .$$

We hence deduce that Succ occurs if and only if $\hat{\mathrm{H}}\big(\varphi^\times | \varphi^*\big) \neq 0$ for every $k^\times \in \mathcal{K} \setminus \{k^*\}$, which is equivalent to the following event

$$E \ : \ \forall k^\times \in \mathcal{K} \backslash \{k^*\}, \ \exists (i,j), \ 1 \leq i < j \leq N, \text{ s.t. } \varphi^*(x^{(i)}) = \varphi^*(x^{(j)}) \text{ and } \varphi^\times(x^{(i)}) \neq \varphi^\times(x^{(j)}) \ .$$

Note that this event is close but different from the intersection $E_1 \cap E_2$ analyzed for the collision attack. In particular we have $E \Rightarrow E_1 \cap E_2$ but $E_1 \cap E_2 \not\Rightarrow E$. In other words, our collision attack succeeds whenever MIA succeeds but the converse is not true.

Nevertheless, the event E has still a high probability to occur when the parameters are chosen as in our collision attack. For instance, let $n = 2m$ and $N = \Theta(2^{\frac{m}{2}})$. According to the birthday paradox, we have a high probability to get a small number q of collisions $\varphi^*(x^{(i)}) = \varphi^*(x^{(j)})$. The event E does not occur, if for some $k^\times$ we also have $\varphi^\times(x^{(i)}) = \varphi^\times(x^{(j)})$ for all these q collisions, which happens with probability lower than $\left(\frac{2^m-1}{2^n-N}\right)^q \leq \frac{1}{2^{qm}}$. We thus obtain a high probability $\Pr(E) \geq 1 - \frac{|\mathcal{K}|}{2^{qm}}$ and hence we also have $N = O(2^{\frac{m}{2}})$ for MIA.

Improved MIA. We show hereafter that a simple improvement of MIA can make it as successful as our collision attack. We know that for the right key guess we have $\hat{\mathrm{I}}(\varphi^*; \varphi^*) = \hat{\mathrm{H}}(\varphi^*)$. So for a guess k, we know that if $\hat{\mathrm{I}}(\varphi; \varphi^*) \neq \hat{\mathrm{H}}(\varphi)$ then $k \neq k^*$. Our improvement simply consists in setting the score associated to k to 0 whenever such an inequality occurs, that is

$$\delta_k^{\mathrm{mia}} = \begin{cases} \mathrm{H}(\varphi_k(X)) & \text{if } \max_{j \in [T]} \mathrm{I}\big(\varphi_k(X); V_j\big) = \mathrm{H}(\varphi_k(X)), \\ 0 & \text{otherwise.} \end{cases}$$

For this new distinguisher, we still have $\hat{\mathrm{H}}(\varphi^*)$ as score for the right key guess. But for a wrong key guess, we get a zero score whenever

$$\hat{\mathrm{I}}(\varphi^\times; \varphi^*) = \hat{\mathrm{H}}(\varphi^\times) - \hat{\mathrm{H}}(\varphi^* | \varphi^\times) \neq \hat{\mathrm{H}}(\varphi^\times) \quad \Leftrightarrow \quad \hat{\mathrm{H}}(\varphi^* | \varphi^\times) \neq 0 \;.$$

Therefore, for this improved distinguisher, the success occurs if and only if for every $k^\times \in \mathcal{K} \setminus \{k^*\}$ we have either $\hat{\mathrm{H}}(\varphi^\times | \varphi^*) \neq 0$ (as for standard MIA) or $\hat{\mathrm{H}}(\varphi^* | \varphi^\times) \neq 0$. Equivalently, the failure occurs if for one $k^\times \in \mathcal{K} \setminus \{k^*\}$ we have $\hat{\mathrm{H}}(\varphi^\times | \varphi^*) = 0$ and $\hat{\mathrm{H}}(\varphi^* | \varphi^\times) = 0$. This failure event holds if the distribution of the inputs $(x^{(i)})_{i \in [N]}$ among the different output values of φ are *isomorphic* for $k^\times$ and k^* in the sense of the balls-and-containers game introduced in Section 5.3. We deduce that the success of the improved MIA is equivalent to the event E_2 considered in the analysis of the collision attack. In other words, the improved MIA succeeds if and only if our collision attack succeeds, except that the improved MIA does not need the event E_1 to occur. We then have similar success probabilities than in Proposition 3 where $|\mathcal{K}|$ can be replaced by $|\mathcal{K}| - 1$. However, this difference has a negligible impact on the number of traces (which clearly appears while looking at Equation 11) and one can consider that the two attacks have similar trace complexities in our idealized model.

6.3 Practical Attack Experiments

We perform practical experiments for the generic and improved MIA against the NSC variant implementation. We use the same target variable as in our DCA and collision attack experiments, which is a MixColumn output in the first round, turned into a $(16, 8)$-VBF –with a key space of size 2^{16}– by fixing two input bytes (see Section 4.4.2 for details). For each attack, we first perform some preprocessing step in order to speed-up the attack.

Preprocessing. In order to save some computation we detect all the "informationally equivalent" samples in the collected traces, namely indexes i and j for which

$$\hat{\mathrm{H}}(V_i) = \hat{\mathrm{H}}(V_j) = \hat{\mathrm{I}}(V_i \;;\; V_j) \;,$$

or equivalently for which there is a one-to-one relation between the samples $(v_i^{(\ell)})_{\ell\in[N]}$ and $(v_j^{(\ell)})_{\ell\in[N]}$. We then remove these "informational" duplicates (*i.e.* we either drop the column i or the column j in the set of computation traces). This can be done efficiently by grouping the indexes for which $\hat{\mathrm{H}}(V_i)$ has a given value and only computing $\hat{\mathrm{I}}(V_i\ ;\ V_j)$ within each group. This preprocessing allows us to compress the computation traces by a factor 10 (from 1850 samples to 185), which is especially interesting given the large key space $|\mathcal{K}| = 2^{16}$.

Standard MIA. We could recover the two key bytes (ranked first) with standard MIA using 115 traces. As an illustration, we plot the the mutual information traces for the right key guess and for 256 (out of $2^{16} - 1$) in Figure 8. These are obtained using 150 computation traces in order to make the right guess peak clearly visible.

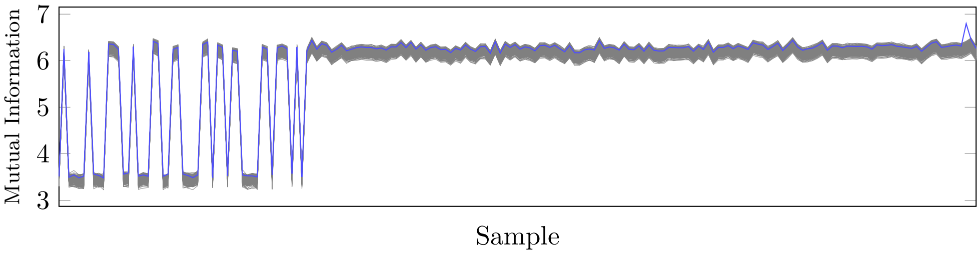

Figure 8: Mutual information traces for the correct key guess (in blue) and 256 (out of $2^{16} - 1$) incorrect key guesses (in gray).

Improved Attack. Our experiments show that the improved attack substantially decreases the number of required traces. Namely, we can recover the right key guess using 70 traces. We observe that only the correct key guess has a positive score, *i.e.* $\hat{\mathrm{H}}(\varphi^*) \approx 5.89$, and all the incorrect key guesses have a score to 0. This is quite similar to our collision attack, which requires 60 traces to recover the same two key bytes on the same implementation.

7 Comparison

In this section, we compare the three attack techniques analyzed in this paper. First of all, they all belong to the same family of *computation analysis* attacks that record computed values during the execution of white-box implementation and then apply side-channel attack techniques. In particular, these attacks are *gray-box* in the sense that they can work with only limited knowledge of the implementation and of the computation traces (*e.g.* one does not need to know the implementation details or the location of the target variables in the traces).

Another apparent similarity is that these three techniques require a non-injective property of the target variable. However we stress that this requirement is intrinsic to collision and mutual information attacks (as already noted in [PR09, BGP+11] for the latter), whereas it is not for DCA. Indeed the necessity of targeting non-injection is implied by the context, namely the presence of random encodings to protect the implementation, and DCA could work without such a requirement in other contexts.

Table 4: Trace complexity of DCA, collision attack (CA) and MIA against internal encodings, where m is the encoding bit-size ($m = 8$ in the NSC variant).

	DCA (Section 4)	**CA** (Section 5)	**MIA** (Section 6)
Theoretical trace complexity	$O\left(2^{2m}\right)$	$O\left(2^{\frac{m}{2}}\right)$	$O\left(2^{\frac{m}{2}}\right)$
Number of traces for NSC variant	1800	60	70

The trace complexities of the three approaches are summarized in Table 4. Notably, compared to DCA, our collision attack and MIA have low trace complexities. Namely, they only require about $2^{\frac{m}{2}}$ traces and are thus very effective at defeating internal encodings. For instance, while using 16-bit encodings would imply a huge code size,[10] our collision attack or improved MIA would still break the implementation with a few hundred traces.

In terms of time complexity, DCA and MIA take $\Theta(T \cdot |\mathcal{K}| \cdot N)$ operations whereas the collision attack takes $\Theta(T \cdot |\mathcal{K}| \cdot N^2)$ operations. Assuming that we have $|\mathcal{K}| = 2^n$ and that we take $n = 2m$ as suggested in our analyses these time complexities become $O(2^{4m} \cdot T)$, $O(2^{3m} \cdot T)$, and $O(2^{2.5m} \cdot T)$ for DCA, CA and MIA respectively. Despite its slightly better asymptotic time complexity, MIA was slower than our collision attack in our practical attack experiments. This would probably not be the case in a setting where more computation traces are required (typically with a higher m).

Finally, we note that our collision attack and MIA are especially suited to attack internal encodings since both attacks rely on the collision behavior of some target variables which is not affected by the application of a random bijection. This explains their superiority over DCA in this context. However, in the presence of different countermeasures (especially affecting the collision behavior) these attacks could fail where a DCA could still succeed. We let a broader comparison in different white-box scenarios as an open topic for future research.

8 Conclusion

In this paper, we have taken one step towards the analysis and improvement of differential computation (gray-box) attacks against white-box implementations. In particular, we have focused on implementations protected with internal encodings and we have conducted a thorough analysis of DCA in this context. Our results formally pinpoint when and why DCA succeeds in defeating internal encodings. In particular, whereas DCA was believed inefficient in breaking byte encodings, we have shown that it actually works for a good choice of the target variable, which was validated through practical attack experiments on publicly available white-box implementations.

Besides, we have introduced a new collision-based DCA-like attack and we have suggested to apply mutual information analysis (MIA) against internally encoded white-box implementation. These two approaches have been theoretically and practically demonstrated very powerful in this context where they achieve a much lower trace complexity than DCA.

Our work suggests that internal encodings, as sole white-box countermeasure, are insufficient to resist DCA-like attacks. Several interesting research directions are raised by our work such as analyzing the level of resistance that can be reached by a combination of countermeasures against this kind of gray-box attacks, and improving these attack techniques, or finding new ones, to address further white-box settings.

[10] With 16-bit encodings, a single encoded table taking two arguments (*e.g.* an encoded XOR) requires $2^{2\times16} \times 2$ bytes which is more than 8 GB.

Acknowledgments

The authors are grateful to Albert Spruyt for helpful discussions. This work was partially done while the second author was visiting Riscure. The second author was supported by European Union's Horizon 2020 research and innovation program under the Marie Skłodowska-Curie grant agreement No. 643161.

References

[BBB$^+$17] Estuardo Alpirez Bock, Joppe W. Bos, Chris Brzuska, Charles Hubain, Wil Michiels, Cristofaro Mune, Eloi Sanfelix Gonzalez, Philippe Teuwen, and Alexander Treff. White-box cryptography: Don't forget about grey box attacks. Cryptology ePrint Archive, Report 2017/355, 2017. https://eprint.iacr.org/2017/355.

[BBMT18] Estuardo Alpirez Bock, Chris Brzuska, Wil Michiels, and Alexander Treff. On the ineffectiveness of internal encodings - revisiting the DCA attack on white-box cryptography. In Bart Preneel and Frederik Vercauteren, editors, *ACNS 18*, volume 10892 of *LNCS*, pages 103–120. Springer, Heidelberg, July 2018.

[BCD06] Julien Bringer, Hervé Chabanne, and Emmanuelle Dottax. Perturbing and protecting a traceable block cipher. In *Communications and Multimedia Security, 10th IFIP TC-6 TC-11 International Conference, CMS 2006, Heraklion, Crete, Greece, October 19-21, 2006, Proceedings*, pages 109–119, 2006.

[BDL97] Dan Boneh, Richard A. DeMillo, and Richard J. Lipton. On the importance of checking cryptographic protocols for faults (extended abstract). In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 37–51. Springer, Heidelberg, May 1997.

[BGEC04] Olivier Billet, Henri Gilbert, and Charaf Ech-Chatbi. Cryptanalysis of a white box AES implementation. In Helena Handschuh and Anwar Hasan, editors, *SAC 2004*, volume 3357 of *LNCS*, pages 227–240. Springer, Heidelberg, August 2004.

[BGP$^+$11] Lejla Batina, Benedikt Gierlichs, Emmanuel Prouff, Matthieu Rivain, François-Xavier Standaert, and Nicolas Veyrat-Charvillon. Mutual information analysis: a comprehensive study. *Journal of Cryptology*, 24(2):269–291, April 2011.

[BHMT16] Joppe W. Bos, Charles Hubain, Wil Michiels, and Philippe Teuwen. Differential computation analysis: Hiding your white-box designs is not enough. In Benedikt Gierlichs and Axel Y. Poschmann, editors, *CHES 2016*, volume 9813 of *LNCS*, pages 215–236. Springer, Heidelberg, August 2016.

[BRVW18] Andrey Bogdanov, Matthieu Rivain, Philip S. Vejre, and Junwei Wang. Higher-order DCA against standard side-channel countermeasures. Cryptology ePrint Archive, Report 2018/869, 2018. https://eprint.iacr.org/2018/869.

[BS97] Eli Biham and Adi Shamir. Differential fault analysis of secret key cryptosystems. In Burton S. Kaliski Jr., editor, *CRYPTO'97*, volume 1294 of *LNCS*, pages 513–525. Springer, Heidelberg, August 1997.

[CEJvO02a] Stanley Chow, Philip A. Eisen, Harold Johnson, and Paul C. van Oorschot. White-box cryptography and an AES implementation. In *Selected Areas in Cryptography, 9th Annual International Workshop, SAC 2002, St. John's, Newfoundland, Canada, August 15-16, 2002. Revised Papers*, pages 250–270, 2002.

[CEJvO02b] Stanley Chow, Philip A. Eisen, Harold Johnson, and Paul C. van Oorschot. A white-box DES implementation for DRM applications. In *Security and Privacy in Digital Rights Management, ACM CCS-9 Workshop, DRM 2002, Washington, DC, USA, November 18, 2002, Revised Papers*, pages 1–15, 2002.

[Cor99] Jean-Sébastien Coron. Resistance against differential power analysis for elliptic curve cryptosystems. In Çetin Kaya Koç and Christof Paar, editors, *CHES'99*, volume 1717 of *LNCS*, pages 292–302. Springer, Heidelberg, August 1999.

[FY$^+$38] Ronald Aylmer Fisher, Frank Yates, et al. Statistical tables for biological, agricultural and medical research. *Statistical tables for biological, agricultural and medical research.*, 1938.

[GBTP08] Benedikt Gierlichs, Lejla Batina, Pim Tuyls, and Bart Preneel. Mutual information analysis. In Elisabeth Oswald and Pankaj Rohatgi, editors, *CHES 2008*, volume 5154 of *LNCS*, pages 426–442. Springer, Heidelberg, August 2008.

[GMQ07] Louis Goubin, Jean-Michel Masereel, and Michaël Quisquater. Cryptanalysis of white box DES implementations. In Carlisle M. Adams, Ali Miri, and Michael J. Wiener, editors, *SAC 2007*, volume 4876 of *LNCS*, pages 278–295. Springer, Heidelberg, August 2007.

[Kar11] Mohamed Karroumi. Protecting white-box AES with dual ciphers. In Kyung Hyune Rhee and DaeHun Nyang, editors, *ICISC 10*, volume 6829 of *LNCS*, pages 278–291. Springer, Heidelberg, December 2011.

[KJJ99] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Michael J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 388–397. Springer, Heidelberg, August 1999.

[Koc96] Paul C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In Neal Koblitz, editor, *CRYPTO'96*, volume 1109 of *LNCS*, pages 104–113. Springer, Heidelberg, August 1996.

[LKK18] Seungkwang Lee, Taesung Kim, and Yousung Kang. A masked white-box cryptographic implementation for protecting against differential computation analysis. *IEEE Trans. Information Forensics and Security*, 13(10):2602–2615, 2018.

[LRM$^+$14] Tancrède Lepoint, Matthieu Rivain, Yoni De Mulder, Peter Roelse, and Bart Preneel. Two attacks on a white-box AES implementation. In Tanja Lange, Kristin Lauter, and Petr Lisonek, editors, *SAC 2013*, volume 8282 of *LNCS*, pages 265–285. Springer, Heidelberg, August 2014.

[Man04] Stefan Mangard. Hardware countermeasures against DPA - A statistical analysis of their effectiveness. In Tatsuaki Okamoto, editor, *CT-RSA 2004*, volume 2964 of *LNCS*, pages 222–235. Springer, Heidelberg, February 2004.

[MGH09] Wil Michiels, Paul Gorissen, and Henk D. L. Hollmann. Cryptanalysis of a generic class of white-box implementations. In Roberto Maria Avanzi, Liam Keliher, and Francesco Sica, editors, *SAC 2008*, volume 5381 of *LNCS*, pages 414–428. Springer, Heidelberg, August 2009.

[MME10] Amir Moradi, Oliver Mischke, and Thomas Eisenbarth. Correlation-enhanced power analysis collision attack. In Stefan Mangard and François-Xavier Standaert, editors, *CHES 2010*, volume 6225 of *LNCS*, pages 125–139. Springer, Heidelberg, August 2010.

[MRP13] Yoni De Mulder, Peter Roelse, and Bart Preneel. Cryptanalysis of the Xiao-Lai white-box AES implementation. In Lars R. Knudsen and Huapeng Wu, editors, *SAC 2012*, volume 7707 of *LNCS*, pages 34–49. Springer, Heidelberg, August 2013.

[Mui13] James A. Muir. A tutorial on white-box AES. Cryptology ePrint Archive, Report 2013/104, 2013. http://eprint.iacr.org/2013/104.

[PQ03] Gilles Piret and Jean-Jacques Quisquater. A differential fault attack technique against SPN structures, with application to the AES and KHAZAD. In Colin D. Walter, Çetin Kaya Koç, and Christof Paar, editors, *CHES 2003*, volume 2779 of *LNCS*, pages 77–88. Springer, Heidelberg, September 2003.

[PR09] Emmanuel Prouff and Matthieu Rivain. Theoretical and practical aspects of mutual information based side channel analysis. In Michel Abdalla, David Pointcheval, Pierre-Alain Fouque, and Damien Vergnaud, editors, *ACNS 09*, volume 5536 of *LNCS*, pages 499–518. Springer, Heidelberg, June 2009.

[SLFP04] Kai Schramm, Gregor Leander, Patrick Felke, and Christof Paar. A collision-attack on AES: Combining side channel- and differential-attack. In Marc Joye and Jean-Jacques Quisquater, editors, *CHES 2004*, volume 3156 of *LNCS*, pages 163–175. Springer, Heidelberg, August 2004.

[Sv99] Adi Shamir and Nicko van Someren. Playing “hide and seek” with stored keys. In Matthew Franklin, editor, *FC’99*, volume 1648 of *LNCS*, pages 118–124. Springer, Heidelberg, February 1999.

[XL09] Y. Xiao and X. Lai. A secure implementation of white-box aes. In *2009 2nd International Conference on Computer Science and its Applications*, pages 1–6, Dec 2009.

A Proof of Proposition 1

Proof. Under our idealized model, the functions $(\varepsilon_j \circ \varphi^* + \varphi_i)_{\varphi \in \{\varphi_k : k \in \mathcal{K}\}}$ are mutually independent, hence their bias, *i.e.* the variables $(Y_k)_{k \in \mathcal{K}}$, are also mutually independent. Moreover, the $(Y_{k^\times})_{k^\times \in \mathcal{K} \setminus \{k^*\}}$ are identically distributed. We can then write:

$$\begin{aligned}
\Pr\big(|Y_{k^*}| > \max_{k^\times} |Y_{k^\times}|\big) &= \sum_{y=-2^n}^{2^n} \Pr\big(Y_{k^*} = y\big) \cdot \Pr\big(|y| > \max_{k^\times} |Y_{k^\times}|\big) \\
&= \sum_{y=-2^n}^{2^n} \Pr\big(Y_{k^*} = y\big) \cdot \prod_{k^\times} \Pr\big(|y| > |Y_{k^\times}|\big) \\
&= \sum_{y=-2^n}^{2^n} \Pr\big(Y_{k^*} = y\big) \cdot \Pr\big(|y| > |Y_{k^\times}|\big)^{|\mathcal{K}|-1} \ .
\end{aligned}$$

We further have that the distributions of Y_{k^*} and $Y_{k^\times}$ (for every $k^\times$) are both symmetric centered in 0, that is $\Pr(Y_k = y) = \Pr(Y_k = -y)$ for every k and y, which gives

$$\begin{aligned}
\Pr\big(|Y_{k^*}| > \max_{k^\times} |Y_{k^\times}|\big) &= 2 \sum_{-2^n \leq y < 0} \Pr\big(Y_{k^*} = y\big) \cdot \Pr\big(|y| > |Y_{k^\times}|\big)^{|\mathcal{K}|-1} \\
&= 2 \sum_{-2^n \leq y < 0} \Pr\big(Y_{k^*} = y\big) \cdot \big(1 - 2 \cdot \Phi_{Y_{k^\times}}(y)\big)^{|\mathcal{K}|-1} \ .
\end{aligned}$$

Using a change of variable $y = 2^{n-m+2}z - 2^n$, we have $\Pr\big(Y_{k^*} = y\big) = \Pr_{\widetilde{\mathcal{HG}(m)}}(z)$ and $\Phi_{Y_{k^\times}}(y) = \Phi_{\widetilde{\mathcal{HG}(n)}}(2^{n-m} \cdot z)$, which finally yields:

$$\Pr\big(|Y_{k^*}| > \max_{k^\times} |Y_{k^\times}|\big) = 2 \sum_{0 \leq z < 2^{m-2}} \Pr_{\widetilde{\mathcal{HG}(m)}}(z) \cdot \big(1 - 2 \cdot \Phi_{\widetilde{\mathcal{HG}(n)}}(2^{n-m} \cdot z)\big)^{|\mathcal{K}|-1} \ .$$

□

B Proof of Proposition 2

Proof. For any $i, j \in [m]$, and $k \in \mathcal{K}$, let $Y_k^j = \mathrm{B}(\varepsilon_j \circ \varphi^* + \varphi_i)$, then

$$\begin{aligned}
\Pr\big(\mathsf{Succ}_\varepsilon\big) &= \Pr\left(\max_{j \in [m]} |Y_{k^*}^j| > \max_{j \in [m], k^\times \in \mathcal{K} \setminus \{k^\times\}} |Y_{k^\times}^j|\right) \\
&= \sum_{y=1}^{2^n} \Pr\left(\max_{j \in [m]} |Y_{k^*}^j| = y\right) \cdot \Pr\left(y > \max_{j, k^\times} |Y_{k^*}^j|\right)
\end{aligned}$$

where

$$\begin{aligned}
\Pr\left(\max_{j \in [m]} |Y_{k^*}^j| = y\right) &= \Pr\Big(\Big\{j : |Y_{k^*}^j| = y\Big\} \bigcap \Big\{j : |Y_{k^*}^j| \leq y\Big\} \neq \varnothing\Big) \\
&= \sum_{\ell=1}^{m} \Pr\Big(\Big|\Big\{j : |Y_{k^*}^j| = y\Big\}\Big| = \ell \bigwedge \Big|\Big\{j : |Y_{k^*}^j| < y\Big\}\Big| = m - \ell\Big) \\
&= \sum_{\ell=1}^{m} \binom{m}{\ell} \cdot \Pr\left(|Y_{k^*}^j| = y\right)^{\ell} \cdot \Pr\left(|Y_{k^*}^j| < y\right)^{m-\ell} \\
&= \sum_{\ell=1}^{m} \binom{m}{\ell} \cdot \Pr\left(Y_{k^*}^j = \pm y\right)^{\ell} \cdot \Pr\left(-y < Y_{k^*}^j < y\right)^{m-\ell}
\end{aligned}$$

and

$$\begin{aligned}\Pr\left(y > \max_{j,k^\times} |Y^j_{k^\times}|\right) &= \prod_{j,k^\times} \Pr\big(y > |Y^j_{k^\times}|\big) = \Pr\big(y > |Y^j_{k^\times}|\big)^{m\cdot(|\mathcal{K}|-1)} \\ &= \big(1 - 2\cdot \Phi_{Y_{k^\times}}(-y)\big)^{m\cdot(|\mathcal{K}|-1)}\end{aligned}$$

according to the mutual independence of the $\big(\varphi_k\big)_{k\in\mathcal{K}}$ and the mutual independence of their coordinate functions in our idealized model.

Using a change of variable $y = 2^n - 2^{n-m+2}z$, we have $\Pr\big(Y^j_{k^*} = \pm y\big) = 2\Pr_{\widetilde{\mathcal{HG}(m)}}(z)$ and $\Pr\big(-y < Y^j_{k^*} < y\big) = 1 - 2\Phi_{\widetilde{\mathcal{HG}(m)}}(z)$ and $\Phi_{Y_{k^\times}}(-y) = \Phi_{\widetilde{\mathcal{HG}(n)}}(2^{n-m}\cdot z)$, which gives

$$\Pr\left(\max_{j\in[m]} |Y^j_{k^*}| = y\right) = \sum_{\ell=1}^{m} \binom{m}{\ell} \cdot \Big(2\Pr_{\widetilde{\mathcal{HG}(m)}}(z)\Big)^{\ell} \cdot \Big(1 - 2\Phi_{\widetilde{\mathcal{HG}(m)}}(z)\Big)^{m-\ell}$$

(denoted by $\mu(z)$ hereafter) and

$$\Pr\left(y > \max_{j,k^\times} |Y^j_{k^\times}|\right) = \Big(1 - 2\cdot \Phi_{\widetilde{\mathcal{HG}(n)}}(2^{n-m}\cdot z)\Big)^{m\cdot(|\mathcal{K}|-1)} \quad .$$

In summary,

$$\Pr\big(\mathsf{Succ}_\varepsilon\big) = \sum_{z=1}^{2^{m-2}} \mu(z) \cdot \Big(1 - 2\cdot \Phi_{\widetilde{\mathcal{HG}(n)}}(2^{n-m}\cdot z)\Big)^{m\cdot(|\mathcal{K}|-1)} \quad .$$

□

C Procedures for DCA Simulation

```
procedure DcaSimulation($\varphi, n, m, N$)
    $k^*, k^\times \leftarrow_\$ \mathcal{K}$
    $i, j \leftarrow_\$ \{1, 2, \cdots, m\}$
    for $k \in \mathcal{K}$ do
        for $x \in \mathbb{F}_2^n$ do
            $T_k(x) \leftarrow \big(\varphi_k(x) \gg (i-1)\big) \,\&\, 1$
        end for
    end for
    $c_1, c_2 \leftarrow 0$
    for $l \leftarrow 1$ to $N$ do
        $\varepsilon \leftarrow_\$ \mathcal{E}$
        for $j' \in [m], x \in \mathbb{F}_2^n$ do
            $E_{j'}(x) \leftarrow \big(\varepsilon \circ \varphi_{k^*}(x) \gg (j'-1)\big) \,\&\, 1$
        end for
        for $j' \in [m], k \in \mathcal{K}$ do
            $B_{k,l,j'} \leftarrow$ Bias($E_{j'}, T_k, n$)
        end for
        if $B_{k^*,l,j} > \max_{k \in \mathcal{K} \setminus \{k^*\}} B_{k,l,j}$ then
            $c_1 \leftarrow c_1 + 1$
        end if
        if $\max_{j'} B_{k^*,l,j'} > \max_{k^\times,j'} B_{k^\times,l,j'}$ then
            $c_2 \leftarrow c_2 + 1$
        end if
    end for
    return $\big(B_{k^*,l,j}, B_{k^\times,l,j}\big)_{1 \le l \le N}, \frac{c_1}{N}, \frac{c_2}{N}$
end procedure
```

```
procedure Bias($T_1, T_2, n$)
    $w \leftarrow 0$
    for $i \leftarrow 1$ to $2^n$ do
        $w \leftarrow w + T_1(i) \oplus T_2(i)$
    end for
    return $2^n - 2 \cdot w$
end procedure
```

Figure 9: DCA simulation procedures for a (n, m) selection function φ with N trails. Here $\mathcal{K} = \mathbb{F}_2^n$, and $\varepsilon \leftarrow_\$ \mathcal{E}$ means randomly sampling a m-bit permutation, which can be efficiently achieved by [FY+38].

D Proof of Lemma 2

Proof. We have $\frac{\gamma \cdot \beta - i \cdot \beta}{\gamma \cdot \beta - i} = 1 - \frac{i - \frac{i}{\beta}}{\gamma - \frac{i}{\beta}}$ and $\frac{i-1}{\gamma-1} < \frac{i - \frac{i}{\beta}}{\gamma - \frac{i}{\beta}}$ (since $i \le \beta$ and $i < \gamma$). We deduce

$$g(\alpha, \beta, \gamma) < \prod_{i=1}^{\alpha-1} \left(1 - \frac{i-1}{\gamma-1}\right) .$$

According to the mean inequality we get

$$\begin{aligned} \sqrt[\alpha-1]{\prod_{i=1}^{\alpha-1} \left(1 - \frac{i-1}{\gamma-1}\right)} <& \frac{1}{\alpha-1} \sum_{i=1}^{\alpha-1} \left(1 - \frac{i-1}{\gamma-1}\right) \\ =& 1 - \frac{\alpha-2}{2\gamma-2} < 1 - \frac{\alpha-2}{2\gamma} < \exp\left(-\frac{\alpha-2}{2\gamma}\right) . \end{aligned}$$

Combing the two above formulas concludes the proof. □

E Information Theory Preliminaries

Entropy. The Shannon *entropy* of a discrete random variable $X \in \mathcal{X}$ measures the *uncertainty* of X. It is defined by the following equation

$$\mathrm{H}(X) = -\sum_{x\in\mathcal{X}} \Pr(X = x) \cdot \log_2 \Pr(X = x).$$

The *joint* entropy of two discrete random variables X and Y expresses the uncertainty of the combination of variables:

$$\mathrm{H}(X, Y) = -\sum_{x\in\mathcal{X},y\in\mathcal{Y}} \Pr(X = x, Y = y) \cdot \log_2 \Pr(X = x, Y = y).$$

The joint entropy satisfies, $\mathrm{H}(X, Y) = \mathrm{H}(Y, X)$ and

$$\max\big(\mathrm{H}(X), \mathrm{H}(Y)\big) \le \mathrm{H}(X, Y) \le \mathrm{H}(X) + \mathrm{H}(Y), \tag{12}$$

where the left equality is reached if and only if (iff) Y is a deterministic function of X, and the right equality occurs iff X and Y are independent. The *conditional* entropy of a random variable X given by another variable Y expresses the uncertainty on X if Y is known:

$$\mathrm{H}(X|Y) = -\sum_{x\in\mathcal{X},y\in\mathcal{Y}} \big(\Pr(X = x, Y = y) \cdot \log_2 \Pr(X = x|Y = y)\big),$$

which satisfying

$$0 \le \mathrm{H}(X|Y) \le \mathrm{H}(X)$$

where both the left equality and the right equality are reached with the same condition of Equation 12.

Mutual Information. The *mutual information* (MI) of two discrete random variables X and Y expresses the dependence between them. It measures the quantity of information one has obtained on X by observing Y. It is defined as

$$\mathrm{I}(X;Y) \quad = \quad -\sum_{x\in\mathcal{X},y\in\mathcal{Y}} \Pr(X = x, Y = y) \cdot \log_2 \left(\frac{\Pr(X = x, Y = y)}{\Pr(X = x) \cdot \Pr(Y = y)}\right).$$

It can also be computed by the Shannon entropy

$$\begin{aligned}\mathrm{I}(X;Y) &= \mathrm{H}(X) - \mathrm{H}(X|Y)\\ &= \mathrm{H}(X) + \mathrm{H}(Y) - \mathrm{H}(X, Y)\\ &= \mathrm{H}(X, Y) + \mathrm{H}(X|Y) - \mathrm{H}(Y|X).\end{aligned}$$

The mutual information satisfies $\mathrm{I}(X;Y) = \mathrm{I}(Y;X)$ and

$$0 \le \mathrm{I}(X;Y) \le \min\big(\mathrm{H}(X), \mathrm{H}(Y)\big)$$