Symbolic analysis of higher-order side channel countermeasures

Elia Bisi, Filippo Melzani, Vittorio Zaccaria

Abstract

In this paper, we deal with the problem of efficiently assessing the higher order vulnerability of a hardware cryptographic circuit. Our main concern is to provide methods that allow a circuit designer to detect early in the design cycle if the implementation of a Boolean-additive masking countermeasure does not hold up to the required protection order. To achieve this goal, we promote the search for vulnerabilities from a statistical problem to a purely symbolical one and then provide a method for reasoning about this new symbolical interpretation. Eventually we show, with a synthetic example, how the proposed conceptual tool can be used for exploring the vulnerability space of a cryptographic primitive.

Index Terms

Embedded systems security, cryptographic implementations, side channel analysis, higher order differential analysis.

I. Introduction

Cryptography's current research trends show that there is an increasing concern about identifying if a side-channel countermeasure is vulnerable to higher-order attacks. The problem is even more amplified as several previous countermeasures [1], [2], which were considered robust at the higher order, have turned out to be vulnerable [3], [4].

In this paper, we introduce a mathematical tool to assess the higher order vulnerability of a hardware cryptographic circuit. The method empowers the circuit designer to detect if the chosen countermeasure, where sensitive data are combined in a Boolean additive way with random masks (e.g., Boolean masking or a *threshold implementation* [5]), is effective up to the desired order. Our overarching goal is to promote the implied statistical reasoning behind the countermeasure into a symbolical one, eventually extending ordinary computer aided design of integrated circuits.

E. Bisi is with the Department of Statistics, University of Warwick, Coventry - UK.
Email: e.bisi@warwick.ac.uk
F. Melzani is with STMicroelectronics, Agrate Brianza - Italy.
Email: filippo.melzani@st.com
V. Zaccaria is with the Department of Electronics, Information and Bioengineering, Politecnico di Milano, Milano - Italy.
Email: vittorio.zaccaria@polimi.it
This project has been partially funded by EU's H2020 grant n. 644052 - HECTOR.

Admittedly, this is neither the first nor the most general proposal addressing such an ambitious goal. A considerable number of recent works tackle the problem from the "information flow" point of view, by recasting it into either *type checking* or *formal proof verification* [6]–[8]. One peculiar characteristic of the most recent works is the introduction of *probabilistic reasoning* to produce formal proofs of cryptographic programs (e.g., EasyCrypt [9]). While these techniques look promising, they require deep expertise in formal software verification and we believe that our approach provides a lightweight alternative that can be readily adopted by a much larger audience of hardware designers. On the other hand, the conventional, and more practical, approach to assessing the vulnerability of a circuit is simulation-based (see [10] for a comprehensive summary of state-of-the-art techniques). The limits of this approach, however, are twofold; firstly, the scalability is critically impacted by the simulation time. Secondly, the reliability of the outcome is very brittle, i.e., one can practically fail to generate the simulation scenario that produces a specific vulnerability. In contrast, our unique approach allows lifting the problem into a symbolic setting, allowing deterministic reasoning and avoiding simulations.

First of all, in Section II we introduce precise definitions of n-th order and uni/multivariate vulnerability and shed some new light on the connection between combining functions and vulnerability itself. In Section III, under some stricter hypotheses, we promote the search for vulnerabilities from a statistical problem to a purely symbolical one. We then provide a method for reasoning about this new symbolical interpretation, by introducing new conditions for vulnerability (i.e., the XOR-condition) and by exploiting a few classical results of linear algebra in the binary field $\mathbb{F}_2$. Eventually, in Section IV we apply our method to analyze a realistic higher order countermeasure.

II. Notation and problem definition

To define the notation used throughout this paper, we follow some common standards [11]. We use calligraphic letters for sets, e.g. $\mathcal{X}$, and capital letters, e.g. X, for random variables. A generic but deterministic value in $\mathcal{X}$ is denoted by lowercase x. A vector in $\mathcal{X}^m$ is denoted by $\mathbf{x} = (x_i)_{i=1}^m$, where $x_i \in \mathcal{X}$ for all $i = 1, \ldots, m$. We will denote by $H(\boldsymbol{x})$ the Hamming weight of a binary string $\boldsymbol{x}$.

If A and B are events, the notations $\mathbb{P}(A)$ and $\mathbb{P}(A \mid B)$ refer to the probability of A and to the conditional probability of A given B. Similarly, if X and Y are random variables, $\mathbb{E}[X]$ and $\mathbb{E}[X \mid Y]$ refer to the expectation of X and to the conditional expectation of X given Y respectively. Finally, the expectation of X given the event that Y takes the deterministic value y is denoted by $\mathbb{E}[X \mid Y = y]$.

A. Univariate vulnerability

In our work, we define as *side-channel* any physically observable outlet that unintentionally leaks some information from a hardware cryptographic primitive. A *side-channel attack* corresponds to a set of queries to a *physical observable* whose aim is to identify the value of a master key/sub-key K of the primitive [12].

Cryptographic primitives may expose, through a side-channel, one or many intermediate Boolean values, which we call *visible variables* and denote by the letter V because they are effectively processed by the hardware. On the other hand, we call *sensitive variables* the values that are deterministic functions of both the master key K and the public input P; we denote them by $S = S(K, P)$ [13] [14]. For the moment, we note that visible variables are not always sensitive themselves.

Information about a visible value V, hence possibly about some sensitive values, can be derived from observations of a (data dependent) leakage [11]:

$$L = \psi(V) + N\,. \tag{1}$$

Here, ψ is a pseudo-Boolean function[1], while N is a Gaussian random variable which is commonly used to account for measurement noise.

Under the assumption that V is actually a sensitive value S, one might consider a *prediction function* $\hat{L}$

$$\hat{L}(k, P) = \hat{\psi}(S(k, P))\,,$$

where k is any possible value of the key and P is a random plain text. A side-channel attack can thus be mounted by using a *distinguisher function*

$$D(L(K, \cdot), \hat{L}(k, \cdot))$$

monotonically related to the statistical dependency between the actual leakage $L(K, P)$ measured with several random plain texts P, and the leakage model value $\hat{L}(k, P)$ computed using the same plain texts [11]. The attack consists in the optimization problem that aims to find the key guess k_g maximizing the distinguisher:

$$k_g = \text{argmax}_k D(L(K, \cdot), \hat{L}(k, \cdot))\,. \tag{2}$$

It is thus natural to make the following definition of vulnerability to side-channel attacks.

Definition 1 (Vulnerability)**.** We say that a leakage L is *vulnerable* (to a side-channel attack) if it is statistically dependent on a sensitive value, i.e.

$$\mathbb{P}(L = l \mid S = s_1) \neq \mathbb{P}(L = l \mid S = s_2) \tag{3}$$

for some $s_1, s_2 \in \mathcal{S}$ and $l \in \mathcal{L}$.

If such a vulnerability exists, then a distinguisher D may be used to mount an attack.

Example 1 (Correlation Power Analysis Attack of AES)**.** In this type of attack [15] [14], the side-channel considered is the power consumption of the device. The assumption is that a visible value V is actually *sensitive*, for example it corresponds to the output of the AES SBOX in the first round:

$$V = S(K, P) = \text{SBOX}(P \oplus K)\,.$$

Assuming that the leakage is dependent on the Hamming weight of V, the attacker can use the prediction function

$$\hat{L}(k, P) = H(S(k, P))\,,$$

and the Pearson's correlation coefficient as a distinguisher to detect the linear statistical dependence between $\hat{L}$ and L and thus derive the best guess for k.

[1]Namely, ψ is a mapping from the Boolean space where V takes values to the real line. It is often defined to be the Hamming weight of the underlying intermediate value V encoded in binary form.

B. Masking and higher order attacks

To safeguard against the vulnerability expressed in eq. (3), a customary solution is to prevent a *sensitive value* S to become *visible*, by splitting S into d shares $V_1, \dots, V_d$, which are actually processed instead. This means that

$$S = V_1 \star V_2 \star \cdots \star V_d \,, \tag{4}$$

where $\star$ is a group operation (usually the bitwise XOR), $V_2, \dots, V_d$ are random uniformly distributed values called *masks*, and V_1 is the *masked variable* defined in such a way that eq. (4) holds. Thus, each of the shares turns out to be statistically independent of S and cannot be used alone to mount an attack.

However, this procedure does not remove any possible vulnerability: if the leakage L involves more than one share, it might still depend on S. This dependence is often revealed by a combining function $C(L)$ of the leakage: if the expected value of $C(L)$ given S is not constant, then an attack can still be mounted. See [16] for an in-depth analysis of available combining functions. In most cases, C is a polynomial of order greater than one, so that the term *higher order attack* is commonly adopted:

Definition 2 (Vulnerability to an n-th order attack)**.** We say that L is *vulnerable to an n-th order attack* if one of its n-th conditional moments given S is not constant, i.e. there exists an n-th degree polynomial $C(L)$ such that

$$\mathbb{E}[C(L) \mid S = s_1] \neq \mathbb{E}[C(L) \mid S = s_2] \tag{5}$$

for some $s_1, s_2 \in \mathcal{S}$.

Example 2 (Second-order univariate CPA)**.** Let us consider the same scenario of Example 1, assuming now that the device uses a first-order countermeasure. The value $S = S(k, P)$ is thus never explicitly processed, but it is replaced by $V_1 = S \oplus M$ and $V_2 = M$, where M is a mask. Suppose that it is possible to observe a leakage that depends on both shares[2]:

$$L(\mathbf{V}) = H(V_1) + H(V_2) + N \,.$$

The expectation $\mathbb{E}[L \mid S]$ does not depend on S, showing that a first order attack fails. However, setting

$$C(L) = (L - \mathbb{E}[L])^2 \,, \tag{6}$$

the second central moment $\mathbb{E}[C(L) \mid S]$ does depend on S. As shown in [16], this combining function can be used, together with the correlation distinguisher, to derive an optimal key guess.

C. Multivariate vulnerability

The concept of vulnerability stated in Definition 1 can be directly extended to the case where the visible variables are spread over several leakages $\mathbf{L} = (L_i)_{i=1}^{l}$. If an attack exploits the information given by a vector of l leakages, we say that such an attack is *l-variate*. Definition 2 can be generalized to the case of multivariate leakages, by considering an n-th degree polynomial $C(L) \colon \mathbb{R}^l \to \mathbb{R}$ *in l variables* as a combining function, so that the two sides of inequality (5) are *mixed* conditional moments.

[2]This corresponds to the case when both shares are processed at the same time by different computational blocks.

Example 3 (Second-order bi-variate CPA)**.** Let us go back to Example 2, assuming now that the implementation processes only one single share at a time instead of both at the same time. We then have two leakages related to the same sensitive variable S:

$$L_1 = H(V_1) + N\,, \qquad L_2 = H(V_2) + N\,,$$

where $V_1 = S \oplus M$ and $V_2 = M$, being M a mask. The second central mixed moment $\mathbb{E}[C(L) \mid S]$ corresponding to the combining function

$$C(\mathbf{L}) = (L_1 - \mathbb{E}[L_1]) \cdot (L_2 - \mathbb{E}[L_2]) \tag{7}$$

depends on S and can be used to mount a second order bivariate attack, as shown in [16].

D. Remarks on moment-based attacks

Formulas (6) and (7) suggest that statistical moments play an important role in pointing out the vulnerability of a circuit. In fact, some scholars have already noticed that vulnerability arises when a difference in conditional distributions can be detected by looking at the statistical moments through suitable polynomial combining functions. Actually, as demonstrated in [16], there exist some non-polynomial functions that work as well, although not as efficiently.

Nevertheless, we can ask ourselves the following question: if no n-th order attack can be mounted for any n, might the leakage be still vulnerable? The answer is fortunately negative, as we shall see. Let us first clearly state our leakage model:

Assumption 1. The components of the leakage vector $\boldsymbol{L} = (L_i)_{i=1}^{l}$ are of the form

$$L_i = \psi_i(\boldsymbol{V}) + N_i\,,$$

where $\psi_1, \ldots, \psi_l$ are pseudo-Boolean functions and $N_1, \ldots, N_l$ are Gaussian random variables independent of each other and independent of any other variable.

Under this assumption, it is possible to prove the following theorem, which answers our question:

Theorem 1. *Under Assumption 1, a leakage vector is vulnerable if and only if it is vulnerable to an n-th order attack for some n.*

Proof. See Appendix. □

It is worth noting that some different attack approach that is more general (such as MIA [14]), or more efficient, in practical scenarios may exist. Yet, theorem 1 ensures that if a vulnerability is present, there must exist a corresponding polynomial combining function. Accordingly, the analysis performed on such polynomial combining functions is sufficient to assess the presence or lack of any vulnerability.

III. A METHOD FOR DETECTING HIGHER ORDER VULNERABILITY

In this section, we introduce a method to detect higher-order leakage vulnerability by directly analyzing the relationship between visible variables and sensitive ones. This could be extremely useful when the designer of a (hardware) countermeasure wants to make sure that the device does not leak any information up to a certain attack order without performing a full-blown statistical analysis at design time.

Throughout this section, for convenience, we consider all the sensitive, mask and visible bits in column vectors $\boldsymbol{S} = (S_i)_{i=1}^{s} \in \mathbb{F}_2^s$, $\boldsymbol{M} = (M_i)_{i=1}^{m} \in \mathbb{F}_2^m$ and $\boldsymbol{V} = (V_i)_{i=1}^{v} \in \mathbb{F}_2^v$.

We now restrict the leakage model described in Assumption 1, by making the following assumption:

Assumption 2. The components of the leakage vector $\boldsymbol{L} = (L_i)_{i=1}^{l}$ are of the form

$$L_i = \sum_{j=1}^{v} c_{i,j} V_j + N_i \qquad \forall i = 1, \ldots, l\,, \tag{8}$$

where $c_{i,j}$'s are real coefficients, and $N_1, \ldots, N_l$ are Gaussian noises independent of each other and of any other variable.

Given that $c_{i,j}$ can assume arbitrary values for each bit, assumption 2 is extremely general. For example, it covers *i)* a one to one mapping between each value of the visible variables and each value of the leakages (e.g. the identity function) — this is the most desirable case for an attacker — and *ii)* the Hamming weight of some binary string when $c_{i,j}$ is 1 iff the j-th visible bit leaks through the i-th leakage (and 0 otherwise).

As we will see in Theorem 2, the assumption above allows connecting the n-th order leakage vulnerability to the n-vulnerability of the visible vector, in the sense specified by the following definition:

Definition 3. We say that $\boldsymbol{V}$ is *vulnerable* if it is not independent of $\boldsymbol{S}$ and n*-vulnerable* if the *minimal* vulnerable subset of visible bits in $\boldsymbol{V}$ is of size n.

We point out that the property of n-vulnerability for a visible vector can be checked, as we will see later on, without resorting to statistical experiments.

Remark 1. We are considering $\boldsymbol{V}$ and $\boldsymbol{S}$ as composed of v and s variables of 1 bit respectively. In our practical application (see Section IV), however, we deal with variables which can be b bits wide (e.g. 8 bits in the AES SBOX). It is straightforward to extend the above definition of n-vulnerability accordingly.

Remark 2. The n-vulnerability indicates the *order* of the corresponding attack only. Once an n-vulnerability has been detected, it is possible to derive the leakages involved by inspecting $c_{i,j}$'s. As we will see in Section III-B, this allows building the combining function; in particular, the attack will be either univariate or multivariate according to the number of leakages involved.

The following theorem explains how n-vulnerability is connected with our overarching goal of detecting leakage vulnerability.

Theorem 2. *If the vector of visible variables is n-vulnerable, then any leakage of type* (8) *(see Assumption 2) is secure against attacks of order lower than n.*

Proof. Let $\boldsymbol{S} \in \mathbb{F}_2^s$, $\boldsymbol{V} \in \mathbb{F}_2^v$ and $\boldsymbol{L} \in \mathbb{R}^l$ be the vector of sensitive variables, an n-vulnerable visible vector and a leakage of type (8) respectively. We need to prove that $\mathbb{E}[C(\boldsymbol{L}) \mid \boldsymbol{S}]$ is constant for any polynomial $C(\boldsymbol{L})$ of degree

$d < n$ in the variables L_i's. By (8), each L_i can be considered as a real polynomial of degree 1 in the variables V_j's ($j = 1, \dots, v$) and N_i. Therefore, $C(\boldsymbol{L})$ can be considered as a polynomial of degree d in the set of variables

$$\mathcal{V} = \{V_j\}_{j=1}^{v} \cup \{N_i\}_{i=1}^{l} ,$$

i.e. as a sum of a certain number r of monomials $C_1(\mathcal{V}), \dots, C_r(\mathcal{V})$ of degree $\leq d$. We then have

$$\mathbb{E}[C(\boldsymbol{L}) \mid \boldsymbol{S}] = \sum_{i=1}^{r} \mathbb{E}[C_i(\mathcal{V}) \mid \boldsymbol{S}] .$$

Each $C_i(\mathcal{V})$ is of degree $< n$, hence it depends on less than n visible bits V_j's. Since $\boldsymbol{V}$ is n-vulnerable by hypothesis, $C_i(\mathcal{V})$ is independent of $\boldsymbol{S}$, hence all the conditional expectations in the sum above are constant. This proves that $\mathbb{E}[C(\boldsymbol{L}) \mid \boldsymbol{S}]$ is constant, as required. □

On the grounds of Theorem 2, it is thus possible to cast the problem of detecting an n-th order leakage vulnerability to the n-vulnerability of the corresponding visible variables.

A. Detecting n-vulnerability

It is now natural to ask ourselves if it is possible to easily detect n-vulnerability. It turns out that, if we restrict ourselves to the case of Boolean-additive masking where visible variables are $\mathbb{F}_2$-sums of sensitive variables and masks, it is possible to devise a symbolic method to detect such a vulnerability. More formally, we assume that visible variables are related to masks and sensitive variables by the following matrix expression in $\mathbb{F}_2$:

$$\boldsymbol{V} = \begin{bmatrix} B & A \end{bmatrix} \cdot \begin{bmatrix} \boldsymbol{M} \\ \boldsymbol{S} \end{bmatrix} = B\boldsymbol{M} \oplus A\boldsymbol{S} , \tag{9}$$

where $A = (a_{i,j})$ and $B = (b_{i,j})$ are matrices with entries in $\mathbb{F}_2$, of size $v \times s$ and $v \times m$ respectively. We will call $C = \begin{bmatrix} B & A \end{bmatrix}$ the *visible variables' matrix*. Thus, V_i turns out to be the $\mathbb{F}_2$-sum of all M_h's such that $b_{i,h} = 1$ and all S_j's such that $a_{i,j} = 1$:

$$V_i = \bigoplus_{h=1}^{m} b_{i,h} M_h \oplus \bigoplus_{j=1}^{s} a_{i,j} S_j \qquad \forall i = 1, \dots, v .$$

In this setting, we define the following condition associated with the matrix structure of C:

Definition 4 (XOR-condition)**.** We say that a visible vector $\boldsymbol{V} \in \mathbb{F}_2^v$ satisfies the XOR-*condition* if there exists a constant row vector $\boldsymbol{\epsilon} = (\epsilon_i)_{i=1}^{v} \in \mathbb{F}_2^v$ such that the product

$$\boldsymbol{\epsilon}\boldsymbol{V} = \bigoplus_{i=1}^{v} \epsilon_i V_i$$

cancels out any mask contribution (i.e. $\boldsymbol{\epsilon} B = \boldsymbol{0}$) and is a non-constant random variable.

Roughly speaking, the XOR-condition holds when there is a combination of visible bits that: i) does not depend on masks and ii) does depend on some sensitive value. Its importance is highlighted by the following theorem, which permits establishing if $\boldsymbol{V}$ is vulnerable:

Theorem 3. *Let $\boldsymbol{S} \in \mathbb{F}_2^s$, $\boldsymbol{M} \in \mathbb{F}_2^m$ and $\boldsymbol{V} = B\boldsymbol{M} \oplus A\boldsymbol{S} \in \mathbb{F}_2^v$ be the sensitive, mask and visible vectors respectively, being A and B deterministic $\mathbb{F}_2$-matrices.*

1) *The* XOR*-condition implies the vulnerability of* $\boldsymbol{V}$.
2) *Assuming that* $\boldsymbol{S}$ *and* $\boldsymbol{M}$ *are independent and* $\boldsymbol{M}$ *is uniformly distributed in* $\mathbb{F}_2^m$*, the vulnerability of* $\boldsymbol{V}$ *implies the* XOR*-condition.*

Proof. See Appendix. □

Remark 3. While the statistical independence of sensitive and mask vectors is a completely natural assumption, we require that *i*) the mask vector be uniformly distributed[3] in $\mathbb{F}_2^m$ and *ii*) mask bits undergo only $\mathbb{F}_2$-linear transformations. While these might seem tight restrictions, we recall that they hold for notable classes of implementations [1] [17], as well as those parts of threshold-based implementations where masks are transformed linearly [5] [18].

We now deduce a straightforward consequence of Theorem 3, and combine it with what we know from Theorem 2:

Corollary 1. Assume that $\boldsymbol{S}$ and $\boldsymbol{M}$ are independent and $\boldsymbol{M}$ is uniformly distributed in $\mathbb{F}_2^m$. Then, a visible vector $\boldsymbol{V}$ is n-vulnerable if and only if n is the minimum integer such that the XOR-condition holds w.r.t. a vector $\boldsymbol{\epsilon}$ with Hamming weight $H(\boldsymbol{\epsilon}) = n$. In this case, any leakage of type (8) is secure against attacks of order lower than n.

The Hamming weight of $\boldsymbol{\epsilon}$ corresponds here to the number of visible variables that $\boldsymbol{\epsilon}$ actually combines.

Example 4. Consider the following visible variables:

$$\begin{aligned}\boldsymbol{V} &= (V_1, V_2, V_3, V_4)\\ &= (S_1 \oplus M_1, S_2 \oplus M_2, S_1 \oplus S_2 \oplus M_1 \oplus M_2, M_1 \oplus M_2)\end{aligned}$$

In this case, we have only three values of $\boldsymbol{\epsilon} \in \mathbb{F}_2^4$ that cancel out the mask contribution:

$$\begin{aligned}(0,0,1,1) \cdot \boldsymbol{V} &= V_3 \oplus V_4 = S_1 \oplus S_2\,,\\ (1,1,1,0) \cdot \boldsymbol{V} &= V_1 \oplus V_2 \oplus V_3 = 0\,,\\ (1,1,0,1) \cdot \boldsymbol{V} &= V_1 \oplus V_2 \oplus V_4 = S_1 \oplus S_2\,.\end{aligned}$$

For any other $\boldsymbol{\epsilon}$, we have that $\boldsymbol{\epsilon V}$ does show mask contributions. Notice however that the XOR-condition does not hold for $\boldsymbol{\epsilon} = (1,1,1,0)$, because $(1,1,1,0) \cdot \boldsymbol{V}$ is constant. Looking at the other two combinations, we see that the XOR-condition holds for $(0,0,1,1)$ and $(1,1,0,1)$ iff $S_1 \oplus S_2$ is non-constant (this is true, for example, when S_1 and S_2 are independent). In this case, we deduce from Theorem 3 that $\boldsymbol{V}$ is vulnerable. More specifically, by Corollary 1, $\boldsymbol{V}$ is 2-vulnerable. A circuit with leakages of type (8) involving these visible variables might then be vulnerable to 2nd order attacks.

Assuming now that all sensitive bits are independent, any $\mathbb{F}_2$-sum of them is non-constant, unless some of them cancel out. In such a case, we can restate Corollary 1 as an easy-to-verify property of the visible variables' matrix:

[3]All the bits used to mask the implementation must take value 0 or 1 with the same probability and independently of each other.

Corollary 2. Assume that all the variables $S_1, \ldots, S_s,\ M_1, \ldots, M_m$ are independent of each other and $\boldsymbol{M}$ is uniformly distributed in $\mathbb{F}_2^m$. Then, $\boldsymbol{V}$ is n-vulnerable iff

$$n = \min\{H(\boldsymbol{\epsilon})\colon \boldsymbol{\epsilon} \in \mathbb{F}_2^v, \boldsymbol{\epsilon}B = \mathbf{0},\ \boldsymbol{\epsilon}A \neq \mathbf{0}\}\,. \tag{10}$$

In other words, one needs to find the row vectors $\boldsymbol{\epsilon}$ that satisfy the (typically under-determined) $\mathbb{F}_2$-linear system $\boldsymbol{\epsilon}B = \mathbf{0}$ but do *not* satisfy $\boldsymbol{\epsilon}A = \mathbf{0}$: these correspond to all and only the vulnerable combinations.

The formalization expressed in (10) is compliant with the problem space of a Satisfiability Modulo Theory solver [19]. In practice, even though this is an exponentially complex problem, we have found that it is even tractable by brute force.

B. Finding a combination function to exploit the vulnerability

Once an n-vulnerable combination ϵ has been found, by looking at the coefficients $c_{i,j}$'s (see Assumption 2) we can build a combining function in the following way:

$$C(\boldsymbol{L}) = \prod_{j=1}^{v} \phi_j(\epsilon_j), \quad \phi_j(\epsilon) = \begin{cases} 1 & \epsilon = 0 \\ (L_i - E[L_i]) & \epsilon = 1 \wedge \exists c_{i,j} = 1 \end{cases} \tag{11}$$

Note that, for each visible variable V_j in the vulnerable combination, there might be more than one i such that $c_{i,j} = 1$, hence the combining function may not be unique.

C. Determining whether the XOR*-condition holds*

Under the same hypotheses of Corollary 2, which we will always assume throughout this subsection, we now wish to relate the vulnerability of a set of visible variables to a simple algebraic property of the visible variables' matrix.

We first summarize how the visible variables' matrix $C = \begin{bmatrix} B & A \end{bmatrix}$ is constructed, knowing the relations among visible, sensitive and masking variables in a circuit T:

- each row i corresponds to a visible variable V_i of T.
- each column j corresponds to either a mask or a sensitive variable of T. Assuming that the circuit T has m masks and s sensitive variables, C will have $m+s$ columns. The first m columns form matrix B and represent masks; the last s columns form matrix A and represent sensitive variables.
- the entry (i, j) is 1 iff the visible variable V_i happens to depend on the mask or sensitive variable associated with column j.

In light of Corollary 2, we can restate the XOR-condition in the form we are now mainly concerned, referring to a property of the visible variables' matrix: *a visible vector is vulnerable iff there exists $\boldsymbol{\epsilon}$ such that $\boldsymbol{\epsilon}C = \boldsymbol{r}$ is of the form*

$$\boldsymbol{r} = [\mathbf{0}_d\ 1 * \cdots *]\ , \qquad d \geq m\,, \tag{12}$$

where the stars may take any binary value. In words, this means that some $\mathbb{F}_2$-linear combination of the rows of C has all mask columns equal to 0 and at least one sensitive column equal to 1.

Interestingly, as Theorem 4 states, we can determine whether a visible variables' matrix exposes some vulnerability by inspecting its reduced row echelon form. Recall first:

Definition 5. The *reduced row echelon form* C_R of a binary matrix C is the unique binary matrix such that:

1) the row spaces[4] of C_R and C coincide;
2) all nonzero rows of C_R are above any rows of all zeros;
3) in each nonzero row of C_R, the first 1 from the left (also called *pivot*) is strictly to the right of the pivot of the row above;
4) each pivot is the only 1 of its own column.

Theorem 4. *A visible vector is vulnerable if and only if the reduced row echelon form C_R (computed in $\mathbb{F}_2$) of the associated visible variables' matrix C has a sensitive pivot column.*

Proof. Assume that C_R has a sensitive pivot column, i.e. it has a row $\boldsymbol{r}$ whose pivot column is sensitive. Such a row is thus of type (12). On the other hand, $\boldsymbol{r}$ is an $\mathbb{F}_2$-linear combination of rows of C, by property 1) of Definition 5. This shows that the associated visible vector is vulnerable.

Conversely, assume that C_R has no sensitive pivot columns. Consider any row $\boldsymbol{r}$ that is the $\mathbb{F}_2$-combination of n nonzero rows of C_R with pivot columns $j_1, \ldots, j_n$. By property 4), $\boldsymbol{r}$ has 1's in all columns $j_1, \ldots, j_n$. Since all pivot columns are mask columns by hypothesis, $\boldsymbol{r}$ is not of type (12). This proves that the row space of C_R, which coincides with the row space of C again by property 1), does not contain any row of type (12). It follows that the associated visible vector is not vulnerable. $\square$

We point out that Theorem 4 itself allows identifying a vulnerability, but not the minimum combination of visible variables that gives rise to such a vulnerability. Nevertheless, it enables to do so with an interesting computational efficiency. In fact, Albrecht *et al.* [20] proposed an algorithm (M4RI) that allows computing reduced echelon forms in $\mathbb{F}_2$ with a complexity of $\Theta(nm \min(m,n)/\log(m))$ for an $n \times m$ matrix.

On the other hand, Theorem 4 also provides an alternative method, besides Corollary 2, for detecting the vulnerability order. Indeed, it is easy to convince oneself that the linear span of all rows of type (12) that appear in C_R turns out to be the set of all rows $\boldsymbol{r}$ of type (12) that can be written as $\boldsymbol{\epsilon}C$ for some $\boldsymbol{\epsilon}$. Therefore:

Corollary 3. A visible vector is n-vulnerable iff

$$n = \min\{H(\boldsymbol{\epsilon})\colon \boldsymbol{\epsilon} \in \mathbb{F}_2^v, \boldsymbol{\epsilon}C = \boldsymbol{r} \text{ for some } \boldsymbol{r} \in \mathcal{R}\}\,,$$

where $\mathcal{R}$ is the linear span of all rows of type (12) that appear in C_R.

Obviously, the less elements $\mathcal{R}$ has, the more efficient the latter method is.

[4]The row space of a matrix is the vector space of all linear combinations of its rows; in our setting, the underlying field is $\mathbb{F}_2$.

Example 5. The visible variables' matrix associated with Example 4 is

$$C = \left[\begin{array}{cc|cc} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{array}\right], \tag{13}$$

where the vertical line divides the submatrix B, corresponding to the masks M_1, M_2, from the submatrix A, corresponding to the sensitive variables S_1, S_2. The reduced row echelon form of C is

$$C_R = \left[\begin{array}{cc|cc} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \end{array}\right]. \tag{14}$$

We can see that the column of S_1 is a pivot column: by Theorem 4, there is some vulnerability. The only row of type (12) in C_R is $\boldsymbol{r} = \left[\begin{array}{cc|cc} 0 & 0 & 1 & 1 \end{array}\right]$, so $\mathcal{R} = \{\boldsymbol{r}\}$. The solutions to $\boldsymbol{\epsilon} C = \boldsymbol{r}$ are $(0,0,1,1)$ and $(1,1,0,1)$, so $\boldsymbol{V}$ is 2-vulnerable by Corollary 3.

IV. A practical application

To show a practical application of our method, we investigate the potential vulnerabilities of the SubBytes step of the first round of a hypothetical AES implementation similar to the one proposed in [1]. In that work, $\mathbb{F}_2$-linear transformations impact both sensitive variables and masks, while non-linear ones are applied only to sensitive variables using pre-computed lookup tables. We recall, on the grounds of remark 3, that this is exactly one of the cases that can be addressed with our method.

Our implementation (see Figure 1) aims at the protection against first and second order attacks through Boolean masking and consists of four masked SBOX blocks which process 4 bytes per cycle; the entire SubBytes takes thus four cycles to complete. As can be seen, 8 independent random bytes (masks) protect the SubBytes stage and we take care of combining the masks into 16 unique pairs to protect each sensitive variable[5].

We also consider the following visible variables (in addition to the ones shown in Figure 1), which are due to the initialization of the SBOX with the corresponding mask values:

$$V_{i+16} = M_i, \qquad i = 1, \ldots, 8\,. \tag{15}$$

A. Detecting the vulnerabilities

To detect the minimum n-vulnerability, we solve the minimization problem in equation (10) (see Corollary 2). Practically, we find that $\boldsymbol{V}$ is not 1-vulnerable nor 2-vulnerable. However, $\boldsymbol{V}$ is 3-vulnerable, because there are 16 vulnerable combinations of three visible variables, all of the form $(S_x \oplus M_y \oplus M_z, M_y, M_z)$. Vulnerable combinations

[5]For example, the mask pair (M_1, M_2) is only used to mask V_1, despite the fact that M_1 and M_2 are reused for other sensitive variables. This is the result of a trade-off between implementation complexity and vulnerability. In fact, one could *i*) reuse the same four mask pairs in each cycle, exposing a second-order vulnerability, or *ii*) use 32 masks as recommended in [1], guaranteeing second-order protection at expense of circuit complexity and amount of random values.

of four visible variables can also be found, e.g. (V_2, V_5, V_6, V_9). In the following subsection, we will explore this finding in more detail through a synthetic attack that exploits the information about such combinations.

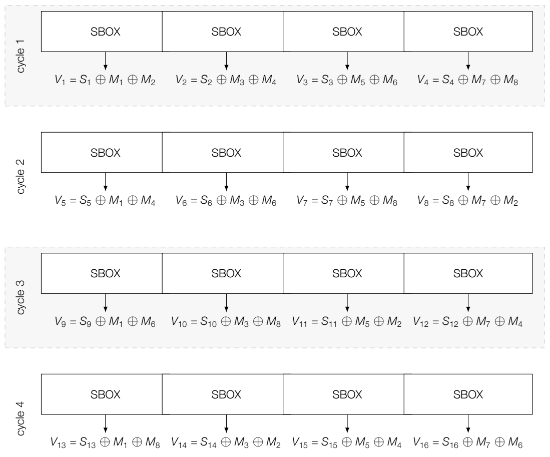
cycle 1
SBOX
SBOX
SBOX
SBOX
$V_1 = S_1 \oplus M_1 \oplus M_2$
$V_2 = S_2 \oplus M_3 \oplus M_4$
$V_3 = S_3 \oplus M_5 \oplus M_6$
$V_4 = S_4 \oplus M_7 \oplus M_8$
cycle 2
SBOX
SBOX
SBOX
SBOX
$V_5 = S_5 \oplus M_1 \oplus M_4$
$V_6 = S_6 \oplus M_3 \oplus M_6$
$V_7 = S_7 \oplus M_5 \oplus M_8$
$V_8 = S_8 \oplus M_7 \oplus M_2$
cycle 3
SBOX
SBOX
SBOX
SBOX
$V_9 = S_9 \oplus M_1 \oplus M_6$
$V_{10} = S_{10} \oplus M_3 \oplus M_8$
$V_{11} = S_{11} \oplus M_5 \oplus M_2$
$V_{12} = S_{12} \oplus M_7 \oplus M_4$
cycle 4
SBOX
SBOX
SBOX
SBOX
$V_{13} = S_{13} \oplus M_1 \oplus M_8$
$V_{14} = S_{14} \oplus M_3 \oplus M_2$
$V_{15} = S_{15} \oplus M_5 \oplus M_4$
$V_{16} = S_{16} \oplus M_7 \oplus M_6$

Figure 1. Cycle by cycle behavior of the considered architecture. In each of the four cycles, we compute four SBOX from four input variables. The output of each SBOX is a sensible variable which is masked with a pair of masks, taking care that each pair is not repeated twice.

B. Mounting the attack

In this section, we mount a 3rd order attack towards $(S_2 \oplus M_3 \oplus M_4, M_3, M_4)$ — one of the configurations found previously — to find the corresponding key K_2. Given Remark 1, we work with variables of 8 bits and attack the SBOX output byte as a whole, by using a commonly adopted procedure [3].

In fact, we assume that each cycle i has a measurable leakage L_i that depends on the Hamming weight of the involved variables (plus normal noise): for $i = 1, 2, 3, 4$,

$$L_i = \sum_{j=4i-3}^{4i} H(V_j) + N_i \,.$$

Moreover, we consider the following initialization leakages:

$$L_{i1} = \sum_{i=0}^{3} H(V_{17+2i}) + N_{i1}, \quad L_{i2} = \sum_{i=0}^{3} H(V_{18+2i}) + N_{i2} \,.$$

A unique feature of our approach is that, given the potentially vulnerable combinations *we can directly derive the leakages they originate from and combine them to extract the key* (see equation (11)). In our case, the vulnerable combinations originate from L_1, L_{i1} and L_{i2} so we build the attack by considering the following product combining function:

$$C(\boldsymbol{L}) = (L_1 - E[L_1])(L_{i1} - E[L_{i1}])(L_{i2} - E[L_{i2}])\,. \tag{16}$$

We sample n leakages from $\boldsymbol{L} = (L_1, L_{i1}, L_{i2})$ by running the synthetic model of the AES primitive with n random plain-texts. We then compute the correlation ρ_{k_2} between the attacker prediction function $H(S_2(k_2, X_2))$

and $C(\boldsymbol{L})$ for every possible key guess k_2, and we choose the k_2 that maximizes it[6].

For each attack, we consider the noise as a normal random variable with mean 0 and standard deviation σ. Both n and σ are thus discrete *factors* of our design of experiments, while each factors' combination corresponds to one experimental unit. In our analysis, factors n and σ assume discrete values (aka *levels*) in the ranges $[20{\cdot}10^3, 100{\cdot}10^3]$ and $[0, 2]$ respectively. For each combination, we measure the corresponding success rate over 50 attacks. Finally, the experiment is repeated 30 times, keeping the plain texts fixed and varying the noise. Figure 2 shows the average success rate for each factor combination and the associated 75% confidence interval. As expected, the ideal case $\sigma = 0$ corresponds to no variation of the success rate.

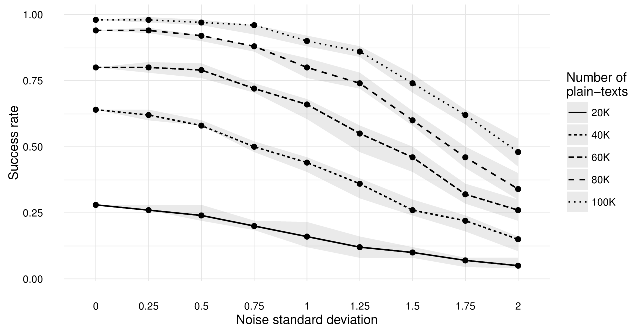

Figure 2. Rate of success in recovering the key, by varying standard deviation of the noise and number of plain texts. Around each black dot, which marks the average success rate, a 75% confidence intervals is plotted as a gray band.

Experiments confirm that keys can be indeed extracted by using our methodology. The success rate clearly increases with the number of experiments; however, even in the ideal case of absence of noise, masking can be effectively defeated only above 60K experiments (success rate $\geq$ 75%). Such a large amount of experiments, however, guarantees a modest ($\sim$ 25%) key extraction success rate when the noise standard deviation is around 2.

As a further example, if we wanted to protect the implementation at the 3rd order, we could have added an additional mask M_9 to all the visible variables. This would eliminate all the vulnerable triplets of the form $(S_x \oplus M_y \oplus M_z, M_y, M_z)$. Note that the countermeasure would still be susceptible to a 4th order attack, e.g. through the quadruple (V_2, V_5, V_6, V_9) and the combining function[7]

$$C(\boldsymbol{L}) = (L_1 - E[L_1])(L_2 - E[L_2])^2(L_3 - E[L_3]).$$

Naturally, to complete such an attack, a suitable function of S_2, S_5, S_6, S_9 should be used as attacker prediction.

V. Conclusions

In this paper, we have proposed a method to detect early in the design cycle if a countermeasure holds up to the required protection order. To do this, we have promoted the search for vulnerabilities to a symbolic analysis by

[6]Following Corollary 11 in [16], and considering our product-based combining function, we have chosen the Hamming weight of the sensitive variable as the attacker prediction function.

[7]Since both V_5 and V_6 are mapped to L_2, the latter is taken two times.

using a theoretically sound approach. To corroborate our findings, we have tested, on a synthetic AES model, to which extent the vulnerabilities found were effectively exploitable.

We conjecture that any system that exposes a suitable leakage function (such as in Assumption 2) and data-flow (such as in eq. (9)) can still be addressed with the presented method. Concretely referring to software implementations, if one can measure the CPU power consumption associated with visible variables being stored in registers or memory then the same analysis could be applied.

Further developments of this work might involve both theoretical aspects — e.g. extending our method to non-linear transformations of masks — and practical aspects — e.g. integrating such an approach in a practical CAD design flow.

VI. Acknowledgments

We thank E. Cagli, G. Bertoni and A. Avalos Pacheco for their assistance and feedback. This project has been partially funded by EU's H2020 (grant n. 644052 - HECTOR).

References

[1] Kai Schramm and Christof Paar. Higher Order Masking of the AES. In *Topics in Cryptology - CT-RSA 2006*, pages 208–225. Springer, February 2006.

[2] Matthieu Rivain and Emmanuel Prouff. Provably Secure Higher-Order Masking of AES. In *Cryptographic Hardware and Embedded Systems - CHES 2010*, pages 413–427. Springer, August 2010.

[3] Jean-Sébastien Coron, Emmanuel Prouff, and Matthieu Rivain. Side Channel Cryptanalysis of a Higher Order Masking Scheme. In *Cryptographic Hardware and Embedded Systems - CHES 2007*, pages 28–44. Springer, September 2007.

[4] Jean-Sébastien Coron, Emmanuel Prouff, Matthieu Rivain, and Thomas Roche. Higher-Order Side Channel Security and Mask Refreshing. In *Smart Card Research and Advanced Applications*, pages 410–424. Springer, Berlin, Heidelberg, July 2014.

[5] Begül Bilgin, Benedikt Gierlichs, Svetla Nikova, Ventzislav Nikov, and Vincent Rijmen. Trade-Offs for Threshold Implementations Illustrated on AES. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 34(7):1188–1200, 2015.

[6] Gilles Barthe, Sonia Belaïd, François Dupressoir, Pierre-Alain Fouque, Benjamin Grégoire, and Pierre-Yves Strub. Verified Proofs of Higher-Order Masking. In *Smart Card Research and Advanced Applications*, pages 457–485. Springer, Berlin, Heidelberg, April 2015.

[7] Ali Galip Bayrak, Francesco Regazzoni, David Novo, and Paolo Ienne. Sleuth: automated verification of software power analysis countermeasures. In *Cryptographic Hardware and Embedded Systems - CHES 2013*, pages 293–310, Berlin, Heidelberg, August 2013. Swiss Federal Institute of Technology, Lausanne, Springer.

[8] Andrew Moss, Elisabeth Oswald, Dan Page, and Michael Tunstall. Compiler Assisted Masking. In *Cryptographic Hardware and Embedded Systems - CHES 2012*, pages 58–75. Springer, Berlin, Heidelberg, September 2012.

[9] Gilles Barthe, François Dupressoir, Benjamin Grégoire, César Kunz, Benedikt Schmidt, and Pierre-Yves Strub. *EasyCrypt: A Tutorial*, pages 146–166. Springer, Cham, 2014.

[10] Oscar Reparaz. Detecting flawed masking schemes with leakage detection tests. In *Fast Software Encryption - FSE 2016*, pages 204–222, 2016.

[11] Suvadeep Hajra and Debdeep Mukhopadhyay. Reaching the limit of nonprofiling DPA. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 34(6):915–927, 2015.

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

[13] Victor Lomné, Emmanuel Prouff, Matthieu Rivain, Thomas Roche, and Adrian Thillard. How to estimate the success rate of higher-order side-channel attacks. In *Cryptographic Hardware and Embedded Systems - CHES 2014*, pages 35–54. Springer, 2014.

[14] 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, 2010.

[15] Eric Brier, Christophe Clavier, and Francis Olivier. chapter Correlation Power Analysis with a Leakage Model, pages 16–29. Springer, Berlin, Heidelberg, 2004.

[16] Emmanuel Prouff, Matthieu Rivain, and Régis Bevan. Statistical analysis of second order differential power analysis. *IEEE Trans. Computers*, 58(6):799–811, 2009.

[17] Jean-Sébastien Coron. Higher order masking of look-up tables. In *Advances in Cryptology - EUROCRYPT 2014*, pages 441–458, 2014.

[18] Thomas De Cnudde, Begül Bilgin, Oscar Reparaz, Ventzislav Nikov, and Svetla Nikova. Higher-Order Threshold Implementation of the AES S-Box. In *Smart Card Research and Advanced Applications*, pages 259–272. Springer, Cham, March 2016.

[19] Leonardo De Moura and Nikolaj Bjørner. Satisfiability modulo theories: introduction and applications. *Communications of the ACM*, 54(9), September 2011.

[20] Martin R Albrecht, Gregory V Bard, and Clément Pernet. Efficient Dense Gaussian Elimination over the Finite Field with Two Elements. *arXiv.org*, November 2011.

[21] Patrick Billingsley. *Probability and Measure*. John Wiley & sons, 2008.

Appendix

Proof of theorem 1

We first need to recall some classical definitions and results from probability theory [21, §21, §30].

Definition 6 (Moment generating function)**.** The *moment generating function* of a multivariate random vector $\boldsymbol{X} = (X_1, \dots, X_l)$ taking values in $\mathbb{R}^l$ is the function

$$M_{\boldsymbol{X}} \colon \mathbb{R}^l \to \mathbb{R}\,, \qquad M_{\boldsymbol{X}}(\boldsymbol{t}) = \mathbb{E}[e^{\boldsymbol{t}\cdot\boldsymbol{X}}] = \mathbb{E}[e^{t_1X_1+\cdots+t_lX_l}]$$

for all $\boldsymbol{t} = (t_1, \dots, t_l) \in \mathbb{R}^l$.

Since the exponential is a positive function, the expectation is well-defined (although it might be infinite for some $\boldsymbol{t}$). Note that the $M_{\boldsymbol{X}}$ is certainly finite at the origin, since $M_{\boldsymbol{X}}(\boldsymbol{0}) = 1$.

Theorem 5. *If $M_{\boldsymbol{X}}$ is finite in a* neighborhood *of the origin, then the distribution of $\boldsymbol{X}$ is* uniquely *determined by its (mixed) moments. Namely, if $\boldsymbol{Y}$ is another $\mathbb{R}^l$-valued random variable with the same moments as $\boldsymbol{X}$, then $\boldsymbol{X}$ and $\boldsymbol{Y}$ are identically distributed.*

Proof. See [21]. □

Proof of theorem 1. Let $\boldsymbol{S}$ and $\boldsymbol{L}$ be the vectors of sensitive variables and leakages respectively.

We first prove the trivial implication: if a leakage is not vulnerable then it is not vulnerable to n-th order attacks for any n. If we assume that $\boldsymbol{L}$ and $\boldsymbol{S}$ are independent, then $C(\boldsymbol{L})$ and $\boldsymbol{S}$ are independent for any combination function $C \colon \mathbb{R}^l \to \mathbb{R}$. This holds in particular when $C(\boldsymbol{L})$ is any n-th degree polynomial; it follows that if $\boldsymbol{L}$ and $\boldsymbol{S}$ are independent (i.e., $\boldsymbol{L}$ is not vulnerable) then $\boldsymbol{L}$ is not vulnerable to n-th order attacks, for any n.

Conversely, assume that $\boldsymbol{L}$ is not vulnerable to n-th order attacks for any n, i.e.

$$\mathbb{E}[C(\boldsymbol{L}) \,|\, \boldsymbol{S} = \boldsymbol{s_1}] = \mathbb{E}[C(\boldsymbol{L}) \,|\, \boldsymbol{S} = \boldsymbol{s_2}] \qquad \forall \boldsymbol{s_1}, \boldsymbol{s_2}\,, \tag{17}$$

for any polynomial C in l variables. The moment generating function of $\boldsymbol{L}$ given $\boldsymbol{S} = \boldsymbol{s}$ can be written as:

$$\begin{aligned} M_{\boldsymbol{L}|\boldsymbol{S}=\boldsymbol{s}}(\boldsymbol{t}) &= \mathbb{E}[e^{\boldsymbol{t}\cdot\boldsymbol{L}} \mid \boldsymbol{S} = \boldsymbol{s}] \\ &= \mathbb{E}[e^{t_1(\psi_1(\boldsymbol{V})+N_1)+\cdots+t_l(\psi_l(\boldsymbol{V})+N_l)} \mid \boldsymbol{S} = \boldsymbol{s}] \\ &= \mathbb{E}[e^{\boldsymbol{t}\cdot\boldsymbol{\psi}(\boldsymbol{V})} \mid \boldsymbol{S} = \boldsymbol{s}] \cdot \mathbb{E}[e^{t_1 N_1}] \cdots \mathbb{E}[e^{t_l N_l}]\,, \end{aligned}$$

where the latter equality follows from the independence hypotheses of Assumption 1. Note that the vector $\boldsymbol{\psi}$ is bounded in $\mathbb{R}^p$, since its components ψ_i's, being pseudo-Boolean functions, can only take a finite number of values; it follows that $\mathbb{E}[e^{\boldsymbol{t}\cdot\boldsymbol{\psi}(\boldsymbol{V})} \mid \boldsymbol{S} = \boldsymbol{s}]$ is finite for all $\boldsymbol{t}$, being the expectation of a bounded random variable. The expectations $\mathbb{E}[e^{t_i N_i}]$ are also finite for all $\boldsymbol{t}$, being moment generating functions of Gaussian random variables (see [21, Example 21.2]).

It follows that $M_{\boldsymbol{L}|\boldsymbol{S}=\boldsymbol{s}}(\boldsymbol{t})$ is finite for all $\boldsymbol{t}$, so that by theorem 5 the distribution of $\boldsymbol{L}$ given $\boldsymbol{S} = \boldsymbol{s}$ is determined by its moments. On the other hand, by eq. (17), we know that the moments of $\boldsymbol{L}$ given $\boldsymbol{S} = \boldsymbol{s_1}$ equal the moments of $\boldsymbol{L}$ given $\boldsymbol{S} = \boldsymbol{s_2}$. The two distributions are therefore equal, thus proving that $\boldsymbol{L}$ is not vulnerable. □

Proof of Theorem 3

Proof. 1) If the XOR-condition holds, there exists $\boldsymbol{\epsilon} \in \mathbb{F}_2^v$ such that $\boldsymbol{\epsilon V} = \boldsymbol{\epsilon} A\boldsymbol{S}$ cancels out any mask contribution and is non-constant. The random variables $\boldsymbol{\epsilon V}$ and $\boldsymbol{\epsilon} A\boldsymbol{S}$ are therefore equal and non-constant, so they are not independent. We conclude that $\boldsymbol{V}$ and $\boldsymbol{S}$ are not independent either, i.e. $\boldsymbol{V}$ is vulnerable.

2) Assume that $\boldsymbol{S}$ and $\boldsymbol{M}$ are independent, $\boldsymbol{M}$ is uniformly distributed in $\mathbb{F}_2^m$ and the XOR-condition does *not* hold. We need to prove that $\boldsymbol{V}$ is not vulnerable, i.e.

$$\mathbb{P}(\boldsymbol{V} = \boldsymbol{v} \mid \boldsymbol{S} = \boldsymbol{s_1}) = \mathbb{P}(\boldsymbol{V} = \boldsymbol{v} \mid \boldsymbol{S} = \boldsymbol{s_2})$$

for all $\boldsymbol{v} \in \mathbb{F}_2^v$ and $\boldsymbol{s_1}, \boldsymbol{s_2} \in \mathbb{F}_2^s$. The latter equality can be equivalently written as

$$\mathbb{P}(B\boldsymbol{M} = \boldsymbol{v} \oplus A\boldsymbol{s_1}) = \mathbb{P}(B\boldsymbol{M} = \boldsymbol{v} \oplus A\boldsymbol{s_2})\,, \tag{18}$$

since $\boldsymbol{S}$ and $\boldsymbol{M}$ are independent. We can see $B\boldsymbol{M} = \boldsymbol{v} \oplus A\boldsymbol{s_1}$ and $B\boldsymbol{M} = \boldsymbol{v} \oplus A\boldsymbol{s_2}$ as two $\mathbb{F}_2$-linear systems of v equations with m unknowns $M_1, \ldots, M_m$, coefficient matrix B and constant vectors $\boldsymbol{v} \oplus A\boldsymbol{s_1}$ and $\boldsymbol{v} \oplus A\boldsymbol{s_2}$ respectively; we will call them $\boldsymbol{s_1}$-system and $\boldsymbol{s_2}$-system respectively. Since $\boldsymbol{M}$ is uniformly distributed in $\mathbb{F}_2^m$, it suffices to prove that the two systems have the same number of solutions c, so that the common value of (18) will be $2^{-m}c$.

Assume first that the $\boldsymbol{s_1}$-system has no solutions; then by Rouché-Capelli theorem the rank of B is strictly less than the rank of the augmented matrix[8] $(B \mid \boldsymbol{v} \oplus A\boldsymbol{s_1})$. Therefore, there exists a nonzero $\mathbb{F}_2$-linear combination of rows of $(B \mid \boldsymbol{v} \oplus A\boldsymbol{s_1})$ that vanishes if restricted to B; in other terms, there exists a row vector $\boldsymbol{\epsilon} \in \mathbb{F}_2^v$ such that $\boldsymbol{\epsilon} B = (0, \ldots, 0)$ but $\boldsymbol{\epsilon}(\boldsymbol{v} \oplus A\boldsymbol{s_1}) = 1$. We then have

$$\boldsymbol{\epsilon V} = \boldsymbol{\epsilon}(A\boldsymbol{S} \oplus B\boldsymbol{M}) = \boldsymbol{\epsilon} A\boldsymbol{S}\,.$$

[8]Recall that the augmented matrix of a linear system is the matrix obtained by appending the columns of the coefficient matrix and the constant vector.

Since we are supposing that the XOR-condition does not hold, we have that $\boldsymbol{\epsilon V}$ is constant, so $\boldsymbol{\epsilon} A\boldsymbol{s_1} = \boldsymbol{\epsilon} A\boldsymbol{s_2}$. Therefore, $\boldsymbol{\epsilon}(\boldsymbol{v} \oplus A\boldsymbol{s_2}) = \boldsymbol{\epsilon}(\boldsymbol{v} \oplus A\boldsymbol{s_1}) = 1$. This fact, along with $\boldsymbol{\epsilon} B = \boldsymbol{0}$, implies that the rank of B is strictly less than the rank of the augmented matrix $(B \mid \boldsymbol{v} \oplus A\boldsymbol{s_2})$; again by Rouché-Capelli theorem, the $\boldsymbol{s_2}$-system has no solutions. To sum up, if one system is not solvable, then the other one is not solvable either, and (18) holds with both sides vanishing.

Assume now that both systems are solvable: let $\boldsymbol{m}(\boldsymbol{s_1})$ and $\boldsymbol{m}(\boldsymbol{s_2})$ be solutions of the $\boldsymbol{s_1}$- and the $\boldsymbol{s_2}$-system respectively. Then, by standard linear algebra, the solution sets of the two systems are $\boldsymbol{m}(\boldsymbol{s_1}) \oplus \ker(B)$ and $\boldsymbol{m}(\boldsymbol{s_2}) \oplus \ker(B)$ respectively. It follows that both systems have c solutions, where c is the cardinality of $\ker(B)$. Note that, by the rank-nullity theorem, $\dim(\ker(B)) = m - r$, where r is the rank of B, hence $\ker(B)$ has 2^{m-r} elements. In conclusion, if one system is solvable, then the other one is, and (18) holds with both sides equal to $2^{-m}2^{m-r} = 2^{-r}$. $\square$