IACR Transactions on Cryptographic Hardware and Embedded Systems
ISSN 2569-2925, Vol. 2022, No. 1, pp. 461–481. DOI:10.46586/tches.v2022.i1.461-481

A Finer-Grain Analysis of the Leakage (Non) Resilience of OCB

Francesco Berti[1], Shivam Bhasin[2], Jakub Breier[3], Xiaolu Hou[4], Romain Poussier[2], François-Xavier Standaert[1] and Balasz Udvarhelyi[1]

[1] UCLouvain, ICTEAM, Crypto Group, Louvain-la-Neuve, Belgium
[2] Temasek Labs, Nanyang Technological University, Singapore
[3] Silicon Austria Labs, Graz, Austria
[4] Slovak University of Technology, Bratislava, Slovakia
firstname.lastname@uclouvain.be
sbhasin@ntu.edu.sg
jbreier@jbreier.com
houxiaolu.email@gmail.com

Abstract. OCB3 is one of the winners of the CAESAR competition and is among the most popular authenticated encryption schemes. In this paper, we put forward a fine-grain study of its security against side-channel attacks. We start from trivial key recoveries in settings where the mode can be attacked with standard Differential Power Analysis (DPA) against some block cipher calls in its execution (namely, initialization, processing of associated data or last incomplete block and decryption). These attacks imply that at least these parts must be strongly protected thanks to countermeasures like masking. We next show that if these block cipher calls of the mode are protected, practical attacks on the remaining block cipher calls remain possible. A first option is to mount a DPA with unknown inputs. A more efficient option is to mount a DPA that exploits horizontal relations between consecutive input whitening values. It allows trading a significantly reduced data complexity for a higher key guessing complexity and turns out to be the best attack vector in practical experiments performed against an implementation of OCB3 in an ARM Cortex-M0. Eventually, we consider an implementation where all the block cipher calls are protected. We first show that exploiting the leakage of the whitening values requires mounting a Simple Power Analysis (SPA) against linear operations. We then show that despite being more challenging than when applied to non-linear operations, such an SPA remains feasible against 8-bit implementations, leaving its generalization to larger implementations as an interesting open problem. We last describe how recovering the whitening values can lead to strong attacks against the confidentiality and integrity of OCB3. Thanks to this comprehensive analysis, we draw concrete requirements for side-channel resistant implementations of OCB3.

Keywords: OCB · side-channel attacks · horizontal DPA · worst-case SPA · leakage-resilience

1 Introduction

The side-channel cryptanalysis of block ciphers has been a topic of intensive research over the last two decades. One important outcome of these research advances is that if no special care is taken, the implementation of any block cipher can be targeted by a side-channel attack, quite independently of its internal components [MOP08]. For example, to a large extent the methods developed to analyze leaking implementations of the AES apply to most (e.g., lightweight) block ciphers developed afterwards [HPGM16]. Yet, some works indicate that while the structure of a block cipher has limited impact on the side-channel attack vectors that can be turned against it (in part because most modern ciphers follow quite similar design principles), the way the block cipher is used in a mode of operation can have a more significant impact [BBC+20].

Received: 2021-07-15 Accepted: 2021-09-15 Published: 2021-11-19

One of the first examples of such an impact is the first-order Differential Power Analysis (DPA) against the AES in counter mode with unknown initial counter put forward by Jaffe [Jaf07]. It was followed by investigations on HMAC which turned out to be attackable given some additional tweaks as well [MTMM07, BBD+13]. A similar issue of secret unknown constants was observed when trying to attack the MILENAGE algorithm used in 3G/4G communications [LYS+15]. Eventually, attacks against the XTS-AES mode of operation have been discussed in [LFDC19] and Jaffe's attack was recently improved in the context of the NIST CTR_DRBG mode [Mey20].

In parallel, the design of modes of operation specifically tailored to mitigate side-channel attacks have evolved under the umbrella of leakage-resilient cryptography. Many constructions have been proposed for this purpose, leveraging various design ideas. We mention [DP08] for one of the first modes exploiting ephemeral key evolution and [BKP+18] for the introduction of strengthened key and tag generation mechanisms, and we mention TEDT [BGP+20], ISAP [DEM+20] and Spook [BBB+20] as three authenticated encryption schemes based on such ideas.

These research advances have been recently discussed in a comprehensive manner at Crypto 2020 [BBC+20]. The main outcome of this discussion is that for some modes of operation, it is possible to reach high security against side-channel attacks without uniformly protecting all the components with expensive countermeasures, leading to so-called leveled implementations.

In this paper, we are interested in the OCB3 mode of operation which, for simplicity, we denote as OCB [RBBK01]. It is among the most popular solutions for instantiating an authenticated encryption scheme efficiently and it is one of the winners of CAESAR competition. In the aforementioned discussion on modes of operation [BBC+20], OCB is referred to as a "Grade-0" design, where all individual components need protection against DPA. *While such an observation is essentially, correct since OCB uses the same long-term key in all the executions of its underlying block cipher, our research question is whether some finer-grain modeling would allow reaching a more balanced view?* In other words, do all the computations involved in an execution of OCB require the same level of security against side-channel analysis or can we identify different types of DPAs against different computations within OCB, or even parts of the computations that only require Simple Power Analysis (SPA) resistance, leading to some possibility of finer-grain leveling in its secure implementation? We contribute to this question by exhibiting several attack vectors against various implementations of OCB, where its most sensitive computations are gradually protected against DPA, and by discussing the complexity of these different attacks.

For this purpose, we first highlight that trivial DPAs can be mounted against the initialization, the processing of associated data, the processing of a final incomplete message block and the decryption of OCB. Preventing these attacks consequently requires a strong and uniform protection of all the block cipher calls within OCB. Note that throughout this paper, the term DPA (resp., SPA) denotes an attack where the number of plaintexts for which the leakage can be observed for a fixed key is under adversarial control (resp., is bounded by design).

We then consider an OCB encryption where the initialization phase is well protected against DPA and discuss attacks that are able to circumvent the secret whitening that this secure initialization implies. A straightforward option is to target the secret whitening as a type of masking scheme and to perform an attack mixing the leakage of two target intermediate values [HTM09, CR17]. Yet, we put forward a more efficient solution that allows attacking such an implementation with a horizontal attack exploiting the relations between consecutive whitening values. It can be viewed as a DPA with a more expensive key guessing strategy, which is well suited to implementations with reasonable noise levels. We show experimentally that it is the best attack vector in the case of an implementation of OCB in an ARM Cortex-M0 and analyze it theoretically.

We finally consider the case where all the block cipher calls of OCB are strongly protected against DPA and the whitening becomes the only target for a side-channel attack. Recovering these whitening values requires performing an SPA against linear operations. We first show experimentally that despite more challenging than similar attacks targeting non-linear operations like [KPP20, BBC+20], such SPAs are feasible against 8-bit implementations. We next argue that they become hard when targeting larger intermediate values (again, even more than when targeting

non-linear operations). We use this observation to confirm the existence of a risk, put forward that (as in general for single-trace SPA) this risk is hard to quantify since it is implementation- and setup-dependent, and leave the generalization of the exhibited SPA to 16-bit or 32-bit implementations as an interesting open problem. We finally show how to exploit the recovery of the whitening values with simple attacks that break the integrity and the confidentiality of OCB.

Based on this finer grain analysis, we conclude that a secure implementation of the OCB encryption requires the strongest DPA protections for its initialization, that the block cipher calls used for the message processing may benefit from slightly weaker countermeasures if only encryption leaks without associated data (since the adversary has to deal with an additional secret whitening to target them), and that the whitening itself becomes a more challenging target in case all the block cipher calls are sufficiently protected. In this last case, we additionally mention that preventing the attack against the whitening layer may be easier as the target attack is an SPA and the operations to protect are linear, making it easier to mask and enabling a mild leveling.

2 Background

2.1 Notations

In this paper, we use capital letters for random variables and small caps for their realizations. We use sans serif font for functions (e.g., F) and calligraphic fonts for sets (e.g., $\mathcal{A}$). We use small bold caps for vectors (e.g., $\mathbf{v}$). We denote the conditional probability of a random variable A given B with $P[A|B]$. We denote as $m^j = \left(m_i^j\right)$ the ith block of the plaintext number j. We denote as $m_i^j = \left(m_i^j(\kappa)\right)$ the κth byte of the ith block of the plaintext number j. We use similar notation for ciphertexts $c_i^j = \left(c_i^j(\kappa)\right)$. When explicit by the context, we will sometimes omit the superscript j if only one plaintext with several blocks is considered. When there is no ambiguity, we will also sometimes omit the byte-indexing for readability reasons. Given a block cipher BC, a master key k and a plaintext block m_i, we denote as $\mathsf{BC}_k(m_i)$ the encryption of m_i under k. We will further denote by k_i the ith round key. We finally denote the concatenation of two values a and b as $a\|b$.

2.2 Template attacks

The main experiments in this study will be done using Template Attacks (TAs) [CRR02]. Such profiled attacks correspond to a strong (ideally the strongest) adversary which is best suited to our main motivation, which is to understand the (ideally worst-case) security of the different parts of OCB. Let $\mathbf{t}$ denote a leakage measured on a cryptographic device that manipulates a target intermediate value $v = \mathsf{F}(m, k)$ associated to a known plaintext byte m and a secret key byte k. In a TA, the adversary first uses a vector of profiling traces $\mathbf{t}_p$ in order to estimate a leakage model, next denoted as $\hat{P}_{\text{model}}[t \mid m, k]$. The profiling traces are typically obtained by measuring a device that is similar to the target under control of the adversary. Next, during the online phase, the adversary uses a vector of new attack traces $\mathbf{t}_a$ from the plaintext vector $\mathbf{m}_a$ obtained by measuring the target device to compute the probability of each key byte guess k as follows:

$$P[k \mid \mathbf{t}_a, \mathbf{m}_a] = \frac{\hat{P}_{\text{model}}[\mathbf{t}_a \mid \mathbf{m}_a, k] \cdot P[k]}{\hat{P}_{\text{model}}[\mathbf{t}_a \mid \mathbf{m}_a]} = \frac{\prod_{t \in \mathbf{t}_a} \hat{P}_{\text{model}}[t \mid m, k]}{\sum_{k_*} \prod_{t \in \mathbf{t}_a} \left[\hat{P}_{\text{model}}[t \mid m, k^*]\right]}. \quad (1)$$

The result of the attack is a vector $\mathbf{p}$ containing the probability of each key byte guess. In the following, we will use Gaussian estimations for the leakage probability density function (PDF). The PDF will thus be estimated by computing mean vectors $\boldsymbol{\mu}$ and covariance matrices $\boldsymbol{\Sigma}$. In addition, we systematically leveraged a dimensionality reduction to combine several informative leakage samples, namely the Linear Discriminant Analysis (LDA) described in [SA08].

2.3 Information theoretic and security metrics

We exploit two information theoretic metrics in our discussions. The Signal-to-Noise Ratio (SNR) is used to identify points-of-interest in the leakage traces. The mutual information is used to discuss the link between a DPA with unknown plaintexts and a DPA against a masked implementation.

Signal-to-noise ratio [Man04]. For a leakage sample L and an a-bit variable X, the SNR is a measure of the (first-order) information leaked. For all values $i \in [0, 2^a - 1]$ that X can take, a leakage set of $j \in [0, b]$ samples of L is acquired and is stored in a vector $\mathbf{s}$. The SNR is then computed as in Equation 2, where E and Var denote the sample mean and variance:

$$\mathrm{SNR}(L, X) = \frac{\mathsf{Var}_j(\mathsf{E}_j(\mathbf{s}))}{\mathsf{E}_i(\mathsf{Var}_j(\mathbf{s}))}. \tag{2}$$

Mutual information [SMY09]. For a leakage sample L and an a-bit variable X, the Mutual Information (MI) computes how many bits of information can be learned about X on average when observing a realization of L. It is computed as per Equation 3:

$$\mathrm{MI}(L, X) = a + \sum_x P[x] \sum_l P[l|x] \cdot \mathsf{log}_2 P[x|l] \cdot \tag{3}$$

We will also use the two security metrics put forward in [SMY09] to evaluate our attacks. First, the Success Rate (SR) is the probability that an attack returns a vector $\mathbf{p}$ such that the correct key byte is ranked first. We will use it to evaluate the DPAs in Section 5 which allow perfect key recoveries. Second, the guessing entropy is the average key rank of the key. We will use it (applied to full keys after rank estimation [PSG16]) to evaluate the more challenging SPA of Section 6.

2.4 OCB mode of operation

The core OCB encryption is depicted in Figure 1 (ignoring the associated data).

A message m consisting of several blocks m_i is encrypted using a block cipher BC to produce a ciphertext block. In this paper, we focus on the case where OCB is instantiated with AES-128, that we denote OCB-AES. First, a value δ_0 is initialized with a user input *nonce*. Using δ_0, several values δ_i are then computed using a function Inc such that $\delta_i = \mathsf{Inc}(\delta_{i-1})$. The δ_0 initialization and Inc function are detailed next. Each δ_i is added to the corresponding plaintext block m_i before its passed to the block cipher. Finally, the output of the block cipher is again added to δ_i in order to produce the ciphertext block c_i. To obtain the last ciphertext block c_ℓ, if m_ℓ is full (i.e., $|m_\ell| = 128$), OCB proceeds as for a normal message block and if m_ℓ is not full, the first $|m_\ell|$ bits of m_ℓ are XORed with $\mathsf{BC}_k(\delta_{\ell-1} \oplus l_*)$, where the l_* value is defined below.

For authentication, a tag τ is computed. It is produced using the checksum of the message $checksum := m_\ell || 10^* \oplus \bigoplus_{i=1}^{\ell-1} m_i$. It then evaluates $\tau = \mathsf{BC}_k\left(checksum \oplus \delta_\ell \oplus l_\$\right) \oplus sum_{AD}$ (with the $l_\$$ value defined below). Finally, The first $|\tau|$ bits of the output are used as a tag.

The associated data is processed in a similar way as the plaintext in Figure 1, replacing the plaintext blocks by the associated data ones. However, in the plaintext case, the initial value δ_0 depends on the encryption of the nonce. In the associated data case, δ_0 is always initialized to 0. The subsequent δ_i values are updated using the same function Inc as for the plaintext case. The output of all the encryption blocks are XORed together, resulting in the value sum_{AD}.

The description of the initialization function Init to compute δ_0 is given in Algorithm 1. As we can see, this processing mainly depends on two components. The first one is the user's input nonce being processed in a known manner to compute the intermediate value top. Second, the value top

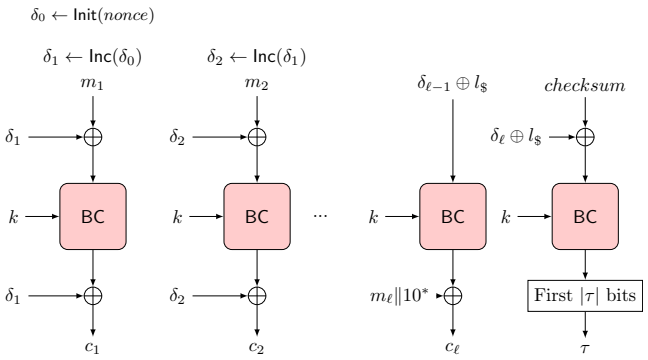

Figure 1: OCB encryption mode (when the last message block is not full).

is used as an input to the block cipher with the master key to produce the value $ktop$. We use the notation $\leftsquigarrow_L$ to highlight the operations for which we will exploit leakages.

The description of the update function Inc is given in Algorithm 2, where $\mathsf{double}(x)$ denotes the multiplication of x by two over $\mathbb{F}_{128}$, and $ntz(x)$ denotes the number of trailing zeros in the base two decomposition of x. First, a value $l_0 = 4 \times \mathsf{BC}_k(0)$ is computed. Then, starting from the previously initialized value δ_0, we compute $\delta_i = \delta_{i-1} \oplus l_{ntz(i)}$ for $m > 0$.

Algorithm 1 Initialization of δ_0 for OCB.

$nonce_2 \leftarrow 0^{127-|nonce|}1\ nonce$
$top \leftarrow nonce_2 \wedge 1^{122}0^6$
$bottom \leftarrow nonce_2 \wedge 0^{122}1^6$
$ktop \leftsquigarrow_L \mathsf{BC}_K(top)$
$stretch \leftarrow ktop \,\|\, (ktop \oplus (ktop \ll 8))$
$\delta_0 \leftarrow (stretch \ll bottom))\,[1..128]$

Algorithm 2 Offset updating for OCB.

$l_* \leftarrow \mathsf{BC}_k(0)$
$l_\$ \leftarrow \mathsf{double}(l_*)$
$l_0 \leftarrow \mathsf{double}(l_\$)$
$l_i \leftarrow \mathsf{double}(l_{i-1})$, for $i \geq 2$
$\delta_1 \leftsquigarrow_L \delta_0 \oplus l_0$
$\delta_2 \leftsquigarrow_L \delta_1 \oplus l_1$
$\delta_3 \leftsquigarrow_L \delta_2 \oplus l_0$
$\delta_4 \leftsquigarrow_L \delta_3 \oplus l_2$
...
$\delta_i \leftsquigarrow_L \delta_{i-1} \oplus l_{ntz(i)}$

3 Experimental setup

In order to validate the results presented in this study, we performed practical experiments for which we now detail the setup. We first discuss the underlying implementation and the associated available leakages. As depicted in Figure 1, the input of the ith block cipher execution is $m_i \oplus \delta_i$. Moreover, as shown in Algorithm 2, we have $\delta_i = \delta_{i-1} \oplus l_{ntz(i)}$. As a result, one can rewrite the ith input of each AES execution such that it always depends on δ_0 and its corresponding sums of l_i. The resulting equations are shown in Algorithm 3. In addition, at the beginning of the AES

Algorithm 3 Rewriting of OCB block cipher inputs depending on δ_0.

$m_1 \oplus \delta_1 = m_1 \oplus \delta_0 \oplus l_0$
$m_2 \oplus \delta_2 = m_2 \oplus \delta_0 \oplus l_0 \oplus l_1$
$m_3 \oplus \delta_3 = m_3 \oplus \delta_0 \oplus l_1$
$m_4 \oplus \delta_4 = m_4 \oplus \delta_0 \oplus l_1 \oplus l_2$
...

execution, the first round key k_0 is XORed to the state and we thus have the initial state equal to $m_i \oplus \delta_i \oplus k_0$. Putting things together, the initial state of each AES execution can be divived in three parts: a known message m_i, a fixed unknown constant $k_i \oplus \delta_0$, and a sum of l_i values that varies depending on the block index j. Concretely, we considered an implementation where all the l_i values are precomputed. It starts by computing δ_0 which is stored in some variable. This variable is then updated by adding $l_{ntz(i)}$ when processing the ith plaintext block. The result is finally passed to the AES state input along with the plaintext block m_i. Overall, and as depicted in Figure 1 and Algorithms 1 and 2, the adversary is provided with leakages on the following operations:

1. Leakages corresponding to the initialization block cipher call $ktop \leftsquigarrow_L \mathsf{BC}_k(top)$.
2. Leakages on the computations and updates $\delta_i \leftsquigarrow_L \delta_{i-1} \oplus l_{ntz(i)}$.
3. Leakages corresponding to the block cipher calls processing the plaintext: $\mathsf{BC}_k(\delta_i \oplus m_i)$

We implemented AES-OCB in an ARM Cortex-M0 microcontroller. The C code used for this purpose was designed to take advantage of 32-bit operations whenever possible, relies on table lookups for the S-box executions and does not include side-channel countermeasures. Our target implementation ignores the processing of the associated data which, as will be shown in Section 4, leads to a trivial attack. We collected measurements using a Picoscope 5244D oscilloscope sampling at 500MSamples/s. We used a shunt resistor of 6 Ohms to measure the current consumption of the target, an STM32F0308 Discovery board. Two small modifications were performed on the board. First, a crystal oscillator was added to the board to provide a stable clock source for our measurements. Second, decoupling capacitors were desoldered. The target device was set to run at its maximum clock frequency of 48MHz. We set up a trigger signal to synchronize our traces so that the ith trace contains the leakages on the update of δ_i and the block cipher call $\mathsf{BC}_k(\delta_i \oplus m_i)$. We measured 800k traces with random inputs for profiling and 1.6M traces for attacks. Our profiling set contains the encryption of 800k plaintexts of one block and our set of attack traces corresponds to 20 plaintexts, each of 80,000 blocks of 128bit.

4 Trivial side-channel attacks

While the OCB mode of operation does not claim any side-channel resistance, attacking a mode of operation might differ from the known/chosen plaintext/ciphertext DPAs against its underlying block cipher. As the latter is usually the one considered in the side-channel attack literature, we first exhibit trivial attacks which can be reduced to that block cipher DPA case.

DPA against the OCB initialization: The leakages associated to the initialization $ktop \leftsquigarrow_L \mathsf{BC}_k(top)$ lead to a trivial first-order DPA attack against the AES block cipher. Indeed, we can see that the initialization procedure (Algorithm 1) performs the encryption of the value top using the master key. As top only depends on the nonce, it can thus be known or chosen by the adversary. As a result, the side-channel security of a fully unprotected implementation of OCB-AES is as low as the security of an unprotected AES in a known or chosen plaintext scenario.

DPA against the OCB decryption: In the decryption scenario, the adversary is able to query OCB-AES with the same nonce several times. In that case, δ_0 and thus δ_1 can be fixed for several ciphertext queries. That scenario again leads to a first-order known or chosen ciphertext DPA against the AES block cipher. For this purpose, the adversary targets the first decryption block c_0^j in two steps. First, a DPA on the external round can be applied, where the key material considered is equal to $\delta_1 \oplus k$. Once recovered, the adversary can predict the value of the state at the next round, which can subsequently be attacked with another DPA, leading to a master key recovery. Note that this second step can be performed using the same set of traces as for the first step.

DPA against the processing of the associated data: As explained in Section 2.4, the whitening value δ_1 is fixed for the associated data processing. As a result, a two-step DPA following exactly the same methodology as for the previous decryption attack can be mounted.

DPA against incomplete messages: As also detailed in Section 2.4, the encryption of the last message block is processed differently if incomplete (i.e., it has to be padded). Let us assume that the index of the last block is i. Ignoring the padding, the corresponding ciphertext block c_i is computed as $c_i = m_i \oplus \mathsf{BC}_k(\delta_i)$ with $\delta_i = \delta_{i-1} \oplus l_*$. That is, there is no addition with δ_i before outputting c_i. As a result, the output of the block cipher call $\mathsf{BC}_k(\delta_i)$ can be computed as $c_i = m_i$ (ignoring the padding). This enables a trivial DPA in a known ciphertext scenario.

These trivial attacks lead to the following conclusions. First, in the case of decryption leakages or in the case where associated data must be authenticated, all the block cipher calls in OCB must be strongly protected against DPA. This is unavoidable given the specifications of OCB and therefore we next consider the more interesting case where only encryption leakages are available and no associated data is processed. Second, the trivial attacks also show that even in this case, the initialization block cipher call and the last block encryption (if incomplete) must be strongly protected against DPA as well. As a result, and in order to deepen our analysis of the leakage properties of OCB, we next consider the case where this initialization (and the last message block encryption) are well protected. This will allow us to determine whether the other operations of OCB require the same (strong) DPA protections. For this purpose, we will investigate the best attack vectors against implementations with gradual protection levels.

5 Protection level 1: secure initialization

In this section, we focus on the case where OCB-AES is used for encryption (without associated data) and its initialization (and the last block encryption) are well protected, leaving the adversary with the block cipher calls used for processing the message blocks as main target. We aim to answer the question whether *weaker protections can be sufficient to protect this part of OCB?*

In this respect, it has already been observed in [BBC+20] that even in the case where only encryption leaks, all the block cipher calls of OCB use the same long-term key, and thus would require some protection against DPA. Yet, attacking the OCB encryption differs from a classical attack against its underlying block cipher, due to the presence of whitening values. Our goal is therefore to provide a finer-grain analysis, evaluating the complexity of the best attacks and comparing them with a standard DPA against the AES (as used by the trivial attacks). More precisely, and as shown in Figure 1, each plaintext block m_i is now XORed with δ_i before being passed to the block cipher. OCB does not allow the same nonce to be repeated for two different plaintext (since it makes no misuse-resistance claims). Hence, δ_0 and all δ_i will be different and unknown for each encryption. In order to deal with this more challenging context, we first describe a (so-called) baseline second-order attack against OCB in Section 5.1: it directly deals with the unknown whitening values by exploiting their leakages in combination with the S-box output leakages. Next, Section 5.2 introduces a more efficient first-order attack. It essentially trades a better data complexity (i.e., number of traces to recover the key) for a higher guessing (time) complexity. We show experimentally that it is indeed the best attack vector against our target ARM Cortex-M0 implementation. We conclude the section by discussing the reasons of this improved data complexity and the extent to which it remains the best attack vector for any noise level.

5.1 Baseline DPA attack

As just mentioned, the main challenge when attacking the OCB encryption of a message is to deal with the unknown δ_i values. It implies that the state of the AES after the initial key addition is $\delta_i \oplus m_i \oplus k$, where δ_i is unknown and different for each encryption block.

A first straightforward approach to deal with this problem is to apply a divide-and-conquer second-order DPA. For this purpose, the adversary will use leakages on one byte of both $\delta_i \oplus m_i$ and $\delta_i \oplus m_i \oplus k$. That is, the adversary will query OCB encryptions for several plaintexts and use the leakages corresponding to the processing of plaintext blocks for each of them. More specifically, and reusing the notations of Section 3, we exploit the following two leakages t_0 and t_1 (since the attack is repeated in the same manner for each byte, we omit the byte notation):

1. The first leakage t_0 targets the the input of $\mathsf{BC_k}$ to retrieve information on $\delta_i \oplus m_i$.
2. The second leakage t_1 uses the block encryption $\mathsf{BC}_k(\delta_i \oplus m_i)$ and targets the first-round Sbox output in order to retrieve information on $\delta_i \oplus m_i \oplus k$.

Using the bivariate leakage $\mathbf{t} = (t_0, t_1)$, we then apply the template attack described in Section 2.2, adapted to deal with the unknown δ_i. That is, Equation 4 shows how to compute $\hat{P}_{\text{model}}[\mathbf{t} \mid k]$ in that case. For completeness, Algorithm 4 shows the pseudocode for the baseline attack.

$$\hat{P}_{\text{model}}[\mathbf{t} \mid k] = \sum_{\delta_i \oplus m_i} \hat{P}_{\text{model}}[\mathbf{t} \mid k, \delta_i \oplus m_i] \cdot P[\delta_i \oplus m_i]\,. \tag{4}$$

Algorithm 4 Baseline attack.

$results_b^i \leftarrow$ array of ones s.t. $i \in \{0, ..., 2^8 - 1\}$ and $b \in \{0, ..., 15\}$
for each attack trace **do**
  for each key byte κ **do**
    for each key byte hypothesis $k_\kappa^* \in 0, ..., 2^8 - 1$ **do**
      update $results_\kappa^{k_\kappa^*}$ using equation 1 and 4
    end for
  end for
end for
for each key byte κ **do**
  return $\mathsf{argmax}(results_\kappa)$ as found key byte
end for

5.2 Improved DPA attack

The main drawback of the baseline attack is that it deals with the unknown whitening values as with a kind of light masking scheme. As will be quantified in Section 5.4, this implies a higher data complexity. We now present an improved DPA attack that circumvents this drawback. At a high level, it exploits the possibility to observe the leakage of a single message consisting of many blocks. By exploiting the relations between different l_i values, we can mount a first-order DPA that reduces the baseline attack's data complexity at the cost of a more expensive guessing strategy.

More precisely, the main idea of the improved DPA is to extend the guessing space from the sole key byte of k_0 by including the corresponding bytes of l_i. Using the rewriting of the block cipher inputs shown in Algorithm 3, we can see that if the l_i are known (or guessed), the only remaining unknown is δ_0. As δ_0 is constant for each block cipher call, we can consider it as a "part of the key" so that a first-order attack can be performed. That is, the search space now becomes one byte of $k_0 \oplus \delta_0 = h$ along with the used bytes of l_i. From a guess on these two values, the output of the first round S-box can be guessed and used for a first-order attack.

The attack only requires to use the leakage t_1 corresponding to the output of the first S-box. However, adding l_i to the search space comes at a cost. By rewriting the equations in Algorithm 3, we can see that the first message block manipulates l_0, which in turn adds one byte to our search space, increasing the time complexity from 2^8 to 2^{16}. The next two blocks further manipulate

l_1, which again increases the search space from 2^{16} to 2^{17}. Indeed, since $l_1 = \mathsf{double}(l_0)$, a 9-bit hypothesis on l_0 is sufficient to represent both corresponding bytes of l_0 and l_1. We observe that we can gradually increase the number of blocks we can consider by increasing the number of l_i values in our search space. In general, adding all l_i values from l_0 to l_n, $n \in [0, 120]$, increases the search space from 2^8 (as for the baseline attack) to 2^{16+n}. Finally, since the ith message block processing uses $l_{\mathsf{ntz}(i)}$, adding all l_i values from l_0 to l_n allows us to consider $2^{n+1} - 1$ message blocks for our attack. Upon success, this first step of the attack will return both the byte of $k_0 \oplus \delta_0$ and the corresponding bytes of l_i. By repeating it for each byte, the attacker is provided with the full values of $k_0 \oplus \delta_0$ and l_0. From the knowledge of $k_0 \oplus \delta_0$ and l_0, the attacker can fully predict the state of each block cipher call after the key addition. Therefore, she can mount a standard first-order attack against the second AES round, for which the state is known before the addition of the second round key k_1. Algorithm 5 provides the pseudo code for the whole attack.

Algorithm 5 Improved attack.

1: $results \leftarrow$ array of ones of size 2^{16+n}
2: **for** each attack trace **do**
3: **for** each byte hypothesis h^* on $k_0 \oplus \delta_0$ and each $8 + n$-bit hypothesis l_0^* on l_0 **do**
4: update $results(h^* \| l_0^*)$ using equation (1)
5: **end for**
6: **end for**
7: Rank hypotheses h^*, l_0^* independently according to their probabilities
8: Recover both bytes of $k_0 \oplus \delta_0$ and the corresponding l_0 value
9: Repeat lines 1-8 with adjustment for each bytes of $k_0 \oplus \delta_0$ and l_0
10: Carry out a standard DPA on the second AES round to recover the second round key

5.3 Experimental results

We implemented the baseline and improved attack using the measurements described in Section 3. For both attacks we used the same profiled DPA using LDA-based dimensionality reduction described in Section 2.2. More precisely, we first identified points-of-interest in the leakage traces by computing the SNR of the target intermediate values over time. We then applied LDA on these points-of-interest to concentrate the leakage into a few dimensions. Our experiments used up to 15 dimensions. We depict in Figure 2 the success rates over the full key of our attacks. They illustrate the gains of the improved attack in terms of data complexity and the weak side-channel security of an implementation of OCB-AES that does not protect its plaintext processing block cipher calls.

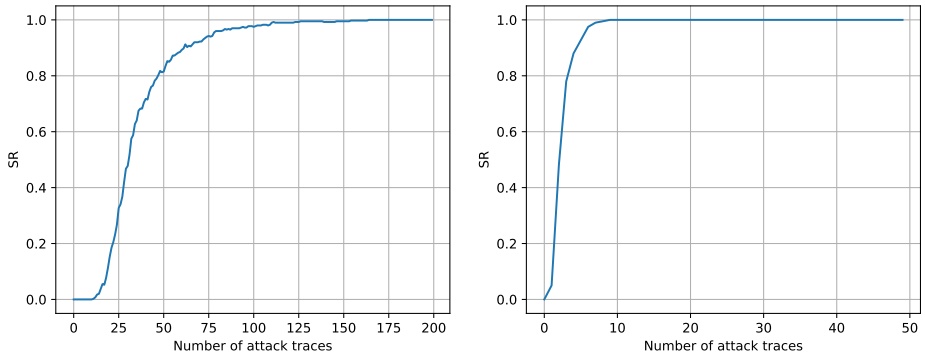

Figure 2: Baseline (left) and improved (right) attacks against an ARM Cortex-M0 implementation.

We note that despite targeting a well-controlled prototype implementation in a permissive (pro-

filed) setting for the adversary, we believe these attacks are sufficient to confirm the need of strong DPA protections for the message processing blocks of OCB. In particular, relaxing the profiled setting is not a significant issue for our improved attack, since adversaries can for example take advantage of an "on-the-fly" regression-based profiling to extract first-order information [DPRS11]. In a similar context as ours, [CR17] also showed how to get rid of the profiling assumption used in [HTM09]. We add an SNR curve illustrating our selection of points-of-interest in Appendix A. It confirms that our implementation leaks as expected for an unprotected implementation (i.e., that our target code did not lead to more leakage than expected for a standard AES implementation).

5.4 Discussion

The previous experimental results show that the improved attack indeed brings concrete benefits over the baseline one. In this section, we complement these practical results with a more theoretical discussion. We start by detailing the reasons of the worse data complexity of the baseline attack by discussing its links with attacks against masked implementations. We then provide a model for predicting the complexity of the improved attack which allows us to conclude that the benefits observed in Section 5.3 should remain stable for a wide range of realistic noise levels.

5.4.1 Baseline attack and masked implementations

There is a tight link between between our baseline attack and attacks against masked implementations [ISW03, RP10]. In case of (Boolean) 2-share masking, any sensitive variable v is decomposed into two values v_1 and v_2 such that $v = v_1 \oplus v_2$. This decomposition ensures that the distribution of any share is independent of the secret. From a security viewpoint, the number of traces required to recover the secret increases therefore exponentially with the number of shares given sufficiently noisy and independent leakages [PR13, DFS19]. A similar noise amplification is observed in our baseline attack (by simply viewing the whitening values as a secret mask).

In the case of an AES implementation protected with masking, the output of the S-box computation is of the form $\mathsf{Sbox}(m \oplus k) \oplus mask$. The attacker is provided with a leakage on that value along with a leakage on the mask. From these two leakages, she can perform a second-order attack (as in our baseline attack). The main difference between our baseline attack and an attack against a Boolean masked implementation lies in the position of the unknown random mask with respect to the S-box computation. That is, in the baseline attack, the unknown δ_i is added to the state inside the S-box computation, and not outside: $\mathsf{Sbox}(m \oplus k \oplus \delta_i)$. We analyzed the mutual information between the leakages and the key byte for these two different attacks using in a simple simulated setting where we assumed the leakage of all the intermediate values in our implementation to be the Hamming weight of these values with additive Gaussian noise. The corresponding results are shown in Figure 3. The X-axis corresponds to the SNR and measures the level of noise. The Y-axis corresponds to the mutual information $\mathrm{MI}(L = (T_1, T_2), K)$, where L denotes the random variable of the leakages and K represent the key random variable. It indicates the worst-case security level. The blue curve corresponds to an unprotected AES, the red curve to a masked implementation with two shares and the yellow curve corresponds to our baseline attack against OCB-AES.

We first observe that both the attack against the 2-share implementation and the baseline attack against AES-OCB have similar slopes in high-noise regimes. This slope indicates the statistical security order of the implementation which is two for these two implementations (vs. one for the unprotected implementation). So the whitening values can indeed be viewed as a impacting security like a 2-share masking scheme. However, we also observe that the curve of the baseline attack against OCB is shifted vertically compared to the one of the 2-share masked AES. This difference is typical of a masking scheme that is not Boolean, for example an Inner Product Masking (IPM). In the context of IPM (see [BFG+17], Figure 3), the algebraic complexity of the encoding makes it slightly harder to exploit leakage even in low-noise regimes. In the OCB-AES case, the unknown δ_i being inside the S-box computation implies a similarly higher algebraic complexity when trying

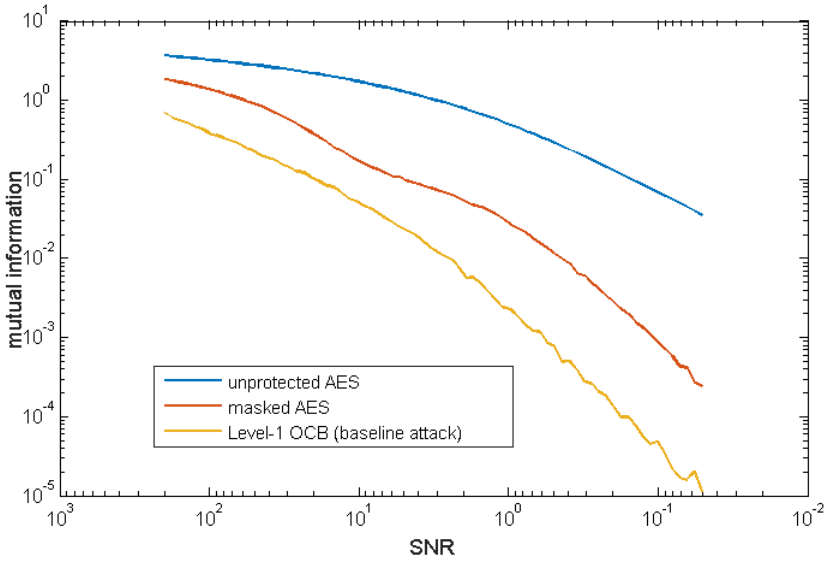

Figure 3: Attacks against masking vs. with unknown inputs: information theoretic analysis.

to recombine the leakages in a second-order attack. These results directly explain the higher data complexity of the baseline attack when compared to the improved one. They further show that the (data complexity) gap between both attacks will increase with the noise level, as does the gap between the MI of the unprotected implementation and the one of the second-order attacks in Figure 3. As a result, the last question to study is whether the baseline attack can sometimes be concretely limited by its higher time complexity, which we discuss next.

5.4.2 Modeling the improved attack

As stated above, in order to observe up to $2^{n+1} - 1$ message blocks, the adversary requires a hypothesis space of a size 2^{16+n}. Since lowering the SNR will increase the number of traces required to attack, a natural question is whether this 2^{16+n} time complexity can become a bottleneck when large n values are needed to recover the key. To answer this question, we derived a model for the attack's data complexity which allows us to determine what is the time complexity of the improved attack for lower SNRs than observed in our experiments. More precisely, we computed a lower bound on the number of traces needed to reach different success rates given different SNRs, for a simple case with a single point-of-interest in each trace. We note that in this setting, we have 256 univariate Gaussian templates corresponding to the leakage of the S-box output.

A series of works have been published for estimating the success rate of DPA attacks (e.g., including [Man04, SPRQ06, Riv08, FLD12, TPR13, LPR+14]). They mostly differ in the distinguishers they consider and the exact assumptions they require. We adopt the notion of the additive distinguisher from [LPR+14] as well as the technique to approximate the score vector (a vector consisting of the distinguisher for each key hypothesis) with a multivariate Gaussian distribution. However, for our analysis, as the number of hypotheses is given by 2^{16+n}, it is difficult to compute the corresponding covariance matrix for the distribution. Instead we apply a formula from [MOP08] to calculate the lower bound for the number of traces assuming the distinguishers are mutually independent. We defer the detailed derivation of this bound to Appendix B.

The results of this bound computed for different success rate values and SNRs can be found in Figure 4. The dashed black line shows number of traces (i.e., encryption blocks) that can be

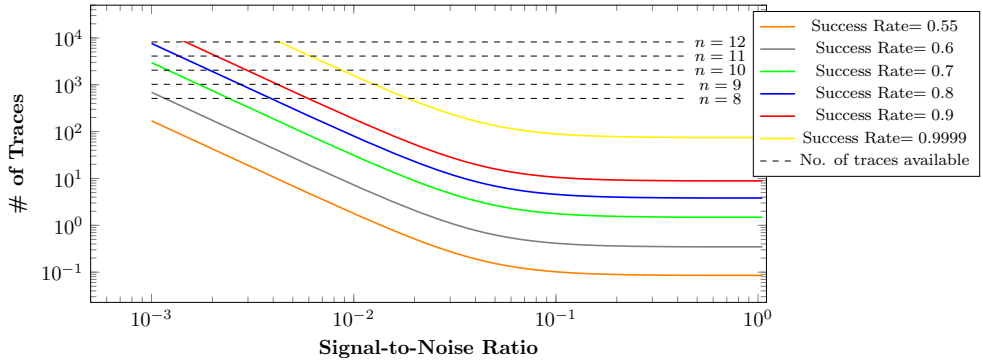

Figure 4: Lower bounds for the # of traces needed to reach different SR of the improved attack. Dashed black line shows number of encryption blocks that can be observed for each n value.

observed for each value of n. The theoretical bounds are consistent with our experimental results. Their main conclusion is that for realistic ranges of SNR, the improved attack will indeed be the method of choice, thanks to its better data complexity and practically reachable time complexity. For example, even the (noisy) unprotected AES hardware implementation from the DPAcontest v2 [rg10] has SNR values between 0.0069 and 0.0096 [PHJL17]. According to Figure 4, it would require a number of traces ranging from 1,000 to 4,000 traces which can be achieved using $n = 11$, leading to a time complexity of 2^{27} which is reasonable even for a standard desktop PC.

6 Protection level 2: secure block cipher calls

Given the negative results of the previous section, we now consider an implementation of OCB-AES where all the block cipher calls are well protected against side-channel attacks. In this case, the adversary is only left with the leakage of the whitening values. We therefore aim to question whether side-channel attacks remain possible. Clearly, key recovery is not an option since OCB's master key is only manipulated by block ciphers. So the question boils down whether it is possible to recover the whitening values, and to exploit them in concrete attacks. We next show that recovering the whitening values requires mounting a SPA against the (linear) operations used to process these values. For this purpose, we describe a worst-case SPA aiming at recovering l_0. We then show experimentally that this attack is applicable to small implementations. As a proof-of-concept, we illustrate it against a weakened implementation where the whitening computations are performed on 8 bits (while our ARM Cortex-M0 implementation naturally allows 32-bit operations). It allows us to illustrate the existence of a risk and to discuss the challenges raised by such attacks and their interpretation in terms of concrete impact. We finally show that if l_0 can be recovered, then strong attacks against the confidentiality and integrity of OCB-AES are possible.

6.1 Worst-case SPA against l_0

In order to recover l_0, we are not able to perform a classical DPA attack. The only available leakages are $\delta_i \leftsquigarrow_L \delta_{i-1} \oplus l_{ntz(i)}$ and $\mathsf{BC}_{\mathsf{input}} \leftsquigarrow_L \delta_i \oplus m_i$. We therefore have to rely on the leakage provided by the loading of $l_{ntz(i)}$ into the registers of the Cortex-M0. Given the function ntz, the value l_0 is used in the whitening of every other plaintext block. Hence, for a plaintext of a given number of blocks i, targeting l_0 in an SPA manner will provide the highest number of attack traces (i.e., $i/2$). The algorithm describing the SPA attack against l_0 is described next.

We additionally investigated the possible improvements of this attack. A first natural option is to exploit the relation between the different l_i's as described in Algorithm 2. This boils down to perform an SPA targeting $l_0, l_1, \ldots$, and recombining their leakages to obtain a more precise

Algorithm 6 SPA against l_0.

$results_b^i \leftarrow$ array of ones s.t. $i \in \{0, ..., 2^8 - 1\}$ and $b \in \{0, ..., 15\}$
for each attack trace l **do**
  for each byte with $\kappa \in \{0, .., 15\}$ **do**
    for each byte hypothesis $h \in \{0, ..., 2^8 - 1\}$ **do**
      update $results_\kappa^h$ using Equation 1
    end for
  end for
end for
return argmax($results_\kappa$) as found key byte

guess on l_0. Another option is to leverage belief propagation to exploit additional leakages on the loading of δ_{i-1}, the computation and storing of δ_i, the plaintext and the computation of the input of BC_k [VGS14]. We observed that these options only lead to marginal gains and therefore do not report them. We posit that the first improvement is of limited effectiveness because the number of traces available to estimate l_{i+1} is half the number of traces to estimate l_i, while the second one is of limited interest because the belief propagation algorithm cannot extract significantly more information than the plain SPA due to the linear nature of the operations targeted.

6.2 Experimental results

We implemented the previous attack against l_0 using the measurements described in Section 3 and the profiled templates with LDA-based dimensionality reduction described in Section 2.2. As in Section 5.3, we preselected points-of-interest based on the SNR and then compressed the informative samples thanks to LDA, keeping up to 15 dimensions. Yet, contrary to the DPA attacks of Section 5.3, we could not recover l_0 in full. The quantiles of the key rank estimated for the full l_0 value thanks to the rank estimation algorithm in [PSG16] are represented in Figure 5.

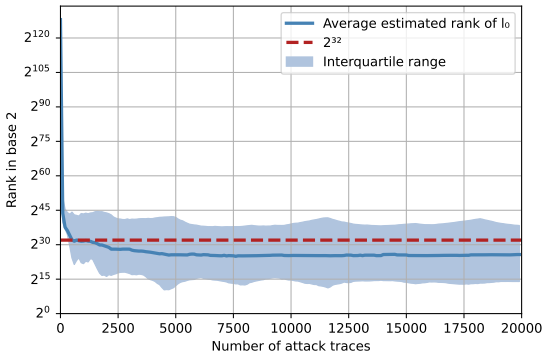

Figure 5: Worst-case attack against l_0 (weakened 8-bit ARM Cortex-M0 implementation).

These results lead to several interesting observations that we detail next:

First, it appears that the SPA against the whitening values is significantly more difficult than a similar (single-trace) attack against a block cipher (or hash function) implementation [KPP20, BBC+20]. In particular, and despite our attacks decrease the key rank significantly, we could not reach a key rank of one for the performed attacks. It implies that exploiting this leakage concretely will require launching the attacks of Section 6.3 and 6.4 multiple times (e.g., 2^{10} times for our easiest target l_0). We posit that this increased difficulty is due to the combined effect of a limited number of leaking operations and the fact that they are linear [Pro05].

Second, and as usual for a SPA, the complexity of the attack depends on the value of l_0. That is, contrary to DPA attacks that are equally difficult for all keys, the efficiency of SPA attacks depends on the target intermediate value [MOS11]. This is reflected in Figure 5 by the quantiles of the key ranks (which are here computed over 20 independent experiments).

Third, and as usual for SPA attacks as well, their worst-case evaluation is more sensitive than the one of a DPA attack. The main reason of this fact is that the efficiency of SPA mostly depends on the (noise-free) deterministic part of the leakage function (while the efficiency of a DPA depends on the SNR or MI). In other words, it depends only on the side-channel signal, which may significantly vary with the implementations and measurement setups. So as discussed in [BMPS21], it is in general a good idea to parameterize implementations conservatively against such attacks.

Overall, the results in this section are therefore conceptually different from the DPA ones in Section 5.3. That is, while these DPA results confirmed the possibility of quite realistic attacks directly justifying a need of strong countermeasures, the SPA results in this section are more of a proof-of-concept showing the existence of a risk. It is an interesting open problem to further investigate how to improve them against larger implementations. For example, applying exactly the strategy of this section to a 16-bit implementation (i.e., limiting the XORs of our ARM Cortex-M0 implementation to 16 bits rather than 8 for Figure 5) already turned out to be much more challenging (i.e., we could not reduce the rank below 2^{80}). Independently of this information extraction question, the next two sections show that recovering l_0 directly leads to concrete attacks against the confidentiality and integrity of OCB. So we first focus on the description of these attacks, before discussing the comparative relevance and impact of these findings in conclusion.

6.3 Attack against the confidentiality of OCB

The attack in this (and the next) section assume that the adversary recovers l_0 via side-channels but, as just mentioned, can be launched for several l_0 candidates in case it is not recovered in full. After the recovery of l_0, the adversary does not need side-channel capabilities anymore: these attacks succeed without any online leakage during the encryption of the challenge plaintext.

In order to simplify the treatment of the attacks, we introduce some additional notations. First, we introduce the set $\mathcal{I}(j)$: it is the set of the indexes of the l_i values used in the computation of δ_j. Similarly, we define the set $\mathcal{I}(j, j')$ as the set of the indexes of the l_i values used in the computation of either δ_j or $\delta_{j'}$ (but not both). That is:

$$\delta_j = \delta_0 \oplus \left(\bigoplus_{i \in \mathcal{I}(j)} l_i \right) \text{ and } \mathcal{I}(j, j') = \mathcal{I}(j) \cup \mathcal{I}(j') \setminus (\mathcal{I}(j) \cap \mathcal{I}(j')) .$$

Finally, we define $l_{(j)} := \bigoplus_{i \in \mathcal{I}(j)} l_i$ and $l_{(j,j')} := \bigoplus_{i \in \mathcal{I}(j,j')} l_i$. Thus, we have:

$$\delta_j = \delta_0 \oplus l_{(j)} \ , \ l_{(j,j')} = l_{(j)} \oplus l_{(j')} \text{ and } \delta_{j'} = \delta_j \oplus l_{(j,j')}.$$

Our attack against the confidentiality of OCB shows that it is possible to distinguish the encryption of a message m from the encryption of another message provided that m has a certain structure. Roughly speaking, the adversary has to distinguish the output of her oracle implemented with OCB_k from a random one. For this output, called the challenge one, she does not receive any leakage. On the other hand, she can query an oracle implemented with OCB_k receiving the outputs and their leakages. So formally, we want to show that the following advantage is not negligible:

$$\left| \Pr[\mathcal{A}^{\mathsf{OCBL}_k, \mathrm{OCB}_k} \Rightarrow 1] - \Pr[\mathcal{A}^{\mathsf{OCBL}_k, \$} \Rightarrow 1] \right| ,$$

where OCBL_k denotes the fact that the adversary has access to the leakage of OCB_k.

The high-level idea of the attack is depicted in Figure 6 (a). It aims at forcing the input of BC_k to be the same in the computation of two different ciphertext blocks. This can be done, for example,

by choosing $m_2 := m_1 \oplus l_{(1)} \oplus l_{(2)} = m_1 \oplus l_{(1,2)}$. In fact, $c_1 = \mathsf{BC}_k(m_1 \oplus l_{(1)} \oplus \delta_0) \oplus l_{(1)} \oplus \delta_0$ while $c_2 = \mathsf{BC}_k(m_2 \oplus l_{(2)} \oplus \delta_0) \oplus l_{(2)} \oplus \delta_0$. Consequently, if $m_2 = m_1 \oplus l_{(1,2)}$, $c_2 = c_1 \oplus l_{(1,2)}$. The attack is specified in Algorithm 7. It can be generalized to any ciphertext block position.

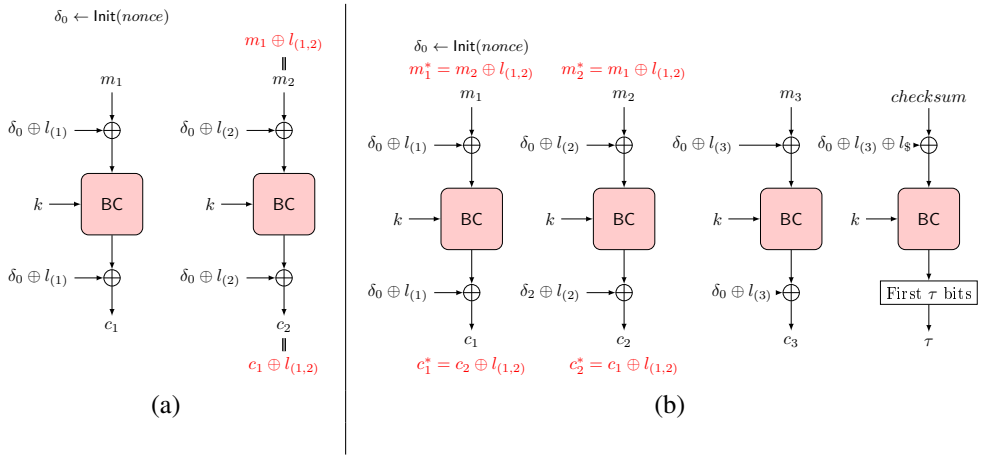

Figure 6: (a) Attack against confidentiality of OCB with side-channel leakage. The adversary asks the encryption of a message $m = (m_1, m_2, m_3, ...)$ with $m_2 = m_1 \oplus l_{1,2}$ with a nonce *nonce*. She can distinguish c_2 from a random block. (b) Attack against authenticity of OCB with side-channel leakage. The adversary asks the encryption of a message $m = (m_1, m_2, m_3)$ with a nonce *nonce*, obtaining $c = (c_1, c_2, c_3, \tau)$. She can output $(nonce, c^*)$ with $c^* = (c_1^*, c_2^*, c_3, \tau)$ as her forgery.

Algorithm 7 Attack against the confidentiality of OCB.

1: $l_1, l_2, ... \leftarrow$ side-channel attack
2: Choose $m_1, m_3, nonce$
3: Choose $m_2 = m_1 \oplus l_{(1,2)}$ and $m = (m_1, m_2, m_3)$
4: Receive $c = (c_1, c_2, c_3, \tau) \leftarrow \mathsf{Enc}_k(nonce, m)$
5: **if** $c_2 = c_1 \oplus l_{(1,2)}$ **then** 1
6: **else** 0
7: **end if**

Note that this attack only applies if the message is composed of more than two message blocks (since the last ciphertext block is computed in a different way). Note also that there exist several definitions of confidentiality in the presence of leakage. For example the one of Barwell et al. [BMOS17] and the one of Guo et al. [GPPS19]. Our attack breaks both definitions.

6.4 Attack against the integrity of OCB

The goal of this second attack, illustrated in Figure 6 (b), is to provide a forgery $(nonce^*, c^*)$ from a ciphertext $c = (C, \tau)$, obtained as the answer of an encryption query on input $(nonce^*, m)$, with the knowledge of the l_is (i.e., formally, to break the ciphertext integrity with leakage definition of [GPPS19]). Its high-level idea is to find a way to force the correct tag τ^* to be the same as τ. We start by observing that the tag depends only on the checksum of a message and on its length. Thus, if $|m| = |m^*|$ and their checksums are the same, their tags are the same if the nonce is the same. To force the checksums to be the same, we use the same idea as for the previous attack against the confidentiality of OCB. In fact, if $m = (m_1, m_2, m_3)$ and (c_1, c_2, c_3, τ) is its encryption, we know that if we replace c_2 with $c_2^* = c_1 \oplus l_{1,2}$, we have encrypted $m_2^* = m_1 \oplus l_{(1,2)}$ instead of m_2. Similarly, if

we replace c_1 with $c_1^* = c_2 \oplus l_{(2,1)}$, we have encrypted $m_1^* = m_2 \oplus l_{(2,1)}$. Finally, we observe that the checksum of the plaintext obtained in the decryption of $C = (c_1, c_2, c_3)$ is the same as the plaintext obtained in the decryption of $C^* = (c_1^*, c_2^*, c_3)$, since $m_1^* \oplus m_2^* = m_1 \oplus l_{(2,1)} \oplus m_2 \oplus l_{(1,2)} = m_2 \oplus m_1$ and $l_{(2,1)} = l_{(1,2)}$. This proves that $(nonce, c^*)$ with $c^* = (C^*, \tau)$ is a valid forgery. The attack is specified in Algorithm 8. It can be easily generalized to longer ciphertexts.

Algorithm 8 Attack against integrity. Forging a fresh and valid ciphertext.

1: $l_1, l_2, ... \leftarrow$ side-channel attack
2: Choose $m_1, m_2, m_3, nonce$
3: Receive $c = (c_1, c_2, c_3, \tau) \leftarrow \mathsf{Enc}_k(nonce, m)$
4: Return $(nonce, c_1 \oplus l_{(2,1)}, c_2 \oplus l_{(2,1)}, c_3, \tau)$

7 Conclusions

In this paper, we studied the resistance of the OCB mode of operation against side-channel attacks. While it was already observed that OCB does not inherently provide side-channel resistance as it uses the same long-term key in all its block cipher calls, we aimed at providing a finer-grain analysis to balance that statement. That is, do all operations in OCB need to be protected and if so, do they require the same level of protection? We answered that question by exhibiting several attacks against OCB for different levels of protections. First, we showed that trivial schoolbook DPA attacks on the underlying block cipher can be mounted against the initialization, the processing of a final incomplete message block, the processing of associated data and the decryption of OCB. Next, assuming a first level of protection preventing the above trivial attacks, we investigated the case where only leakages from the plaintext encryptions are available. In that context, we first showed that one can apply a second-order attack against the secret whitening as one would exploit against a 2-share masked implementation. We also presented an improved (first-order) DPA, decreasing the data complexity of the second-order attack at the cost of a higher (time) key guessing complexity. Experiments and theoretical analysis then confirmed the interest of the improved attack for realistic noise levels. Finally, we considered the case where all block cipher calls are strongly protected against side-channel leakage. We showed that when the whitening values can be recovered by side-channel analysis, it is possible to mount simple attacks that break the integrity and the confidentiality of OCB. Yet, the recovery of these whitening values requires mounting an SPA which is more challenging than the DPAs against the block cipher calls, especially for large computing architectures. From these experiments, we conclude that OCB requires the strongest protections for its initialization, processing of the associated data and the last message block if incomplete. The processing of the message blocks might benefit from a slightly lighter protection (e.g., a masked implementation with one less share). As for the processing of the whitening values, it leads to different intuitions. On the one hand, it is significantly more difficult to target and therefore may require only lighter protections. On the other hand, the worst-case SPAs that can target them are also more difficult to evaluate since quite implementation- and setup-dependent. Given that protecting this (linear) part of OCB thanks to masking is also easier, and since a leakage on them can lead to strong attacks against both the confidentiality and the integrity of the mode, we conclude that it may be a good practice to protect them conservatively as well.

Acknowledgments. This work has been funded in parts by the ERC project 724725 (SWORD). The authors acknowledge partial support from the Singapore National Research Foundation ("SOCure" grant NRF2018NCR-NCR002-0001 – www.green-ic.org/socure). François-Xavier Standaert is a senior associate researcher of the Belgian Fund for Scientific Research.

References

[BBB+20] Davide Bellizia, Francesco Berti, Olivier Bronchain, Gaëtan Cassiers, Sébastien Duval, Chun Guo, Gregor Leander, Gaëtan Leurent, Itamar Levi, Charles Momin, Olivier Pereira, Thomas Peters, François-Xavier Standaert, Balazs Udvarhelyi, and Friedrich Wiemer. Spook: Sponge-based leakage-resistant authenticated encryption with a masked tweakable block cipher. *IACR Trans. Symmetric Cryptol.*, 2020(S1):295–349, 2020.

[BBC+20] Davide Bellizia, Olivier Bronchain, Gaëtan Cassiers, Vincent Grosso, Chun Guo, Charles Momin, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. Mode-level vs. implementation-level physical security in symmetric cryptography - A practical guide through the leakage-resistance jungle. In *CRYPTO (1)*, volume 12170 of *Lecture Notes in Computer Science*, pages 369–400. Springer, 2020.

[BBD+13] Sonia Belaïd, Luk Bettale, Emmanuelle Dottax, Laurie Genelle, and Franck Rondepierre. Differential power analysis of HMAC SHA-2 in the hamming weight model. In *SECRYPT*, pages 230–241. SciTePress, 2013.

[BFG+17] Josep Balasch, Sebastian Faust, Benedikt Gierlichs, Clara Paglialonga, and François-Xavier Standaert. Consolidating inner product masking. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 724–754. Springer, 2017.

[BGP+20] Francesco Berti, Chun Guo, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. Tedt, a leakage-resist AEAD mode for high physical security applications. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2020(1):256–320, 2020.

[BKP+18] Francesco Berti, François Koeune, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. Ciphertext integrity with misuse and leakage: Definition and efficient constructions with symmetric primitives. In *AsiaCCS*, pages 37–50. ACM, 2018.

[BMOS17] Guy Barwell, Daniel P. Martin, Elisabeth Oswald, and Martijn Stam. Authenticated encryption in the face of protocol and side channel leakage. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I*, volume 10624 of *Lecture Notes in Computer Science*, pages 693–723. Springer, 2017.

[BMPS21] Olivier Bronchain, Charles Momin, Thomas Peters, and François-Xavier Standaert. Improved leakage-resistant authenticated encryption based on hardware AES coprocessors. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2021(3):641–676, 2021.

[CR17] Christophe Clavier and Léo Reynaud. Improved blind side-channel analysis by exploitation of joint distributions of leakages. In *CHES*, volume 10529 of *Lecture Notes in Computer Science*, pages 24–44. Springer, 2017.

[CRR02] Suresh Chari, Josyula R Rao, and Pankaj Rohatgi. Template attacks. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 13–28. Springer, 2002.

[DEM+20] Christoph Dobraunig, Maria Eichlseder, Stefan Mangard, Florian Mendel, Bart Mennink, Robert Primas, and Thomas Unterluggauer. Isap v2.0. *IACR Trans. Symmetric Cryptol.*, 2020(S1):390–416, 2020.

[DFS19] Alexandre Duc, Sebastian Faust, and François-Xavier Standaert. Making masking security proofs concrete (or how to evaluate the security of any leaking device), extended version. *J. Cryptol.*, 32(4):1263–1297, 2019.

[DP08] Stefan Dziembowski and Krzysztof Pietrzak. Leakage-resilient cryptography. In *FOCS*, pages 293–302. IEEE Computer Society, 2008.

[DPRS11] Julien Doget, Emmanuel Prouff, Matthieu Rivain, and François-Xavier Standaert. Univariate side channel attacks and leakage modeling. *J. Cryptogr. Eng.*, 1(2):123–144, 2011.

[FLD12] Yunsi Fei, Qiasi Luo, and A Adam Ding. A statistical model for dpa with novel algorithmic confusion analysis. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 233–250. Springer, 2012.

[GPPS19] Chun Guo, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. Authenticated encryption with nonce misuse and physical leakage: Definitions, separation results and first construction - (extended abstract). In Peter Schwabe and Nicolas Thériault, editors, *Progress in Cryptology - LATINCRYPT 2019 - 6th International Conference on Cryptology and Information Security in Latin America, Santiago de Chile, Chile, October 2-4, 2019, Proceedings*, volume 11774 of *Lecture Notes in Computer Science*, pages 150–172. Springer, 2019.

[HPGM16] Annelie Heuser, Stjepan Picek, Sylvain Guilley, and Nele Mentens. Side-channel analysis of lightweight ciphers: Does lightweight equal easy? In *RFIDSec*, volume 10155 of *Lecture Notes in Computer Science*, pages 91–104. Springer, 2016.

[HTM09] Neil Hanley, Michael Tunstall, and William P. Marnane. Unknown plaintext template attacks. In *WISA*, volume 5932 of *Lecture Notes in Computer Science*, pages 148–162. Springer, 2009.

[ISW03] Yuval Ishai, Amit Sahai, and David Wagner. Private circuits: Securing hardware against probing attacks. In *Annual International Cryptology Conference*, pages 463–481. Springer, 2003.

[Jaf07] Joshua Jaffe. A first-order DPA attack against AES in counter mode with unknown initial counter. In *CHES*, volume 4727 of *Lecture Notes in Computer Science*, pages 1–13. Springer, 2007.

[KPP20] Matthias J. Kannwischer, Peter Pessl, and Robert Primas. Single-trace attacks on keccak. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2020(3):243–268, 2020.

[LFDC19] Chao Luo, Yunsi Fei, Aidong Adam Ding, and Pau Closas. Comprehensive side-channel power analysis of XTS-AES. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, 38(12):2191–2200, 2019.

[LPR+14] Victor Lomné, Emmanuel Prouff, Matthieu Rivain, Thomas Roche, and Adrian Thillard. How to estimate the success rate of higher-order side-channel attacks. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 35–54. Springer, 2014.

[LYS+15] Junrong Liu, Yu Yu, François-Xavier Standaert, Zheng Guo, Dawu Gu, Wei Sun, Yijie Ge, and Xinjun Xie. Small tweaks do not help: Differential power analysis of MILENAGE implementations in 3g/4g USIM cards. In *ESORICS (1)*, volume 9326 of *Lecture Notes in Computer Science*, pages 468–480. Springer, 2015.

[Man04] Stefan Mangard. Hardware countermeasures against dpa–a statistical analysis of their effectiveness. In *Cryptographers' Track at the RSA Conference*, pages 222–235. Springer, 2004.

[Mey20] Lauren De Meyer. Recovering the ctr_drbg state in 256 traces. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2020(1):37–65, 2020.

[MOP08] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power analysis attacks: Revealing the secrets of smart cards*, volume 31. Springer Science & Business Media, 2008.

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

[MTMM07] Robert P. McEvoy, Michael Tunstall, Colin C. Murphy, and William P. Marnane. Differential power analysis of HMAC based on sha-2, and countermeasures. In *WISA*, volume 4867 of *Lecture Notes in Computer Science*, pages 317–332. Springer, 2007.

[PHJL17] Stjepan Picek, Annelie Heuser, Alan Jovic, and Axel Legay. Climbing down the hierarchy: hierarchical classification for machine learning side-channel attacks. In *International Conference on Cryptology in Africa*, pages 61–78. Springer, 2017.

[PR13] Emmanuel Prouff and Matthieu Rivain. Masking against side-channel attacks: A formal security proof. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 142–159. Springer, 2013.

[Pro05] Emmanuel Prouff. DPA attacks and s-boxes. In *FSE*, volume 3557 of *Lecture Notes in Computer Science*, pages 424–441. Springer, 2005.

[PSG16] Romain Poussier, François-Xavier Standaert, and Vincent Grosso. Simple key enumeration (and rank estimation) using histograms: An integrated approach. In *CHES*, volume 9813 of *Lecture Notes in Computer Science*, pages 61–81. Springer, 2016.

[RBBK01] Phillip Rogaway, Mihir Bellare, John Black, and Ted Krovetz. OCB: a block-cipher mode of operation for efficient authenticated encryption. In Michael K. Reiter and Pierangela Samarati, editors, *CCS 2001, Proceedings of the 8th ACM Conference on Computer and Communications Security, Philadelphia, Pennsylvania, USA, November 6-8, 2001*, pages 196–205. ACM, 2001.

[rg10] TELECOM ParisTech SEN research group. DPA Contest (2^{nd} edition), 2010.

[Riv08] Matthieu Rivain. On the exact success rate of side channel analysis in the gaussian model. In *International Workshop on Selected Areas in Cryptography*, pages 165–183. Springer, 2008.

[RP10] Matthieu Rivain and Emmanuel Prouff. Provably secure higher-order masking of aes. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 413–427. Springer, 2010.

[SA08] François-Xavier Standaert and Cédric Archambeau. Using subspace-based template attacks to compare and combine power and electromagnetic information leakages. In *CHES*, volume 5154 of *Lecture Notes in Computer Science*, pages 411–425. Springer, 2008.

[SMY09] François-Xavier Standaert, Tal G Malkin, and Moti Yung. A unified framework for the analysis of side-channel key recovery attacks. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 443–461. Springer, 2009.

[SPRQ06] F-X Standaert, Eric Peeters, Gaël Rouvroy, and J-J Quisquater. An overview of power analysis attacks against field programmable gate arrays. *Proceedings of the IEEE*, 94(2):383–394, 2006.

[TPR13] Adrian Thillard, Emmanuel Prouff, and Thomas Roche. Success through confidence: Evaluating the effectiveness of a side-channel attack. In *International Conference on Cryptographic Hardware and Embedded Systems*, pages 21–36. Springer, 2013.

[VGS14] Nicolas Veyrat-Charvillon, Benoît Gérard, and François-Xavier Standaert. Soft analytical side-channel attacks. In *ASIACRYPT (1)*, volume 8873 of *Lecture Notes in Computer Science*, pages 282–296. Springer, 2014.

A Additional SNR figure

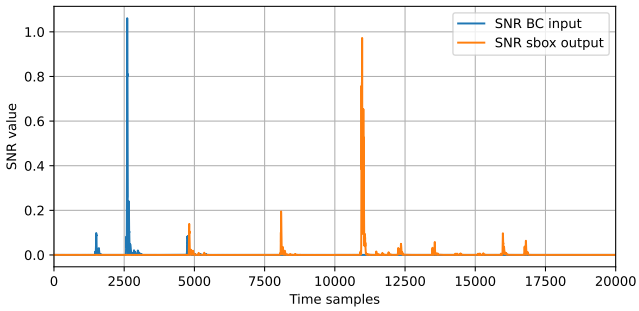

Figure 7: Exemplary SNR curve for the detection of points-of-interest.

B Derivation of the model of Section 5.4.2

In this section we present the detailed calculations for the bound presented in Figure 4. For a fixed n, let $\ell = 2^{n+1} - 1$ denote the number of blocks an attacker can observe, which is the maximum number of traces we have for given n. Let $\mathcal{H}$ denote the set of all possible keys. We consider one key as a concatenated value of the last byte of $\delta_0 \oplus k_0$ and the last $8 + n$ bits of l_0. The attack for other bits follows the similar analysis. Let h_j denote a single key with h_c being the correct key. Recall that $|\mathcal{H}| = 2^{16+n}$. Let T_{x_i,h_j} denote the random variable corresponding to a leakage at encryption of block i with plaintext input m_i and key h_j. For simplicity, we use x_i to denote the last byte of m_i, $m_i(16)$. An observed leakage, denoted by t_i, is a realization of the random variable T_{x_i,h_c}. Let $\mathbf{t}_a := [t_1, t_2, \ldots, t_\ell]$, $\mathbf{m}_a := [x_1, x_2, \ldots, x_\ell]$ denote the vector of leakages and corresponding plaintext bytes respectively. To carry out the template attack, we calculate the probability $P[h_j \mid \mathbf{t}_a, \mathbf{m}_a]$ for each key hypothesis h_j. The guessed key value is then given by the hypothesis achieving the maximum probability. Following equation (1):

$$P[h_j \mid \mathbf{t}_a, \mathbf{m}_a] = \frac{\prod_{i=1}^{\ell} \hat{P}_{\text{model}}[t_i \mid x_i, h_j]}{\sum_{h \in \mathcal{H}} \left(\prod_{i=1}^{\ell} \hat{P}_{\text{model}}[t_i \mid x_i, h] \right)}.$$

Note that the denominator is a constant for any hypothesis h_j. Hence, finding h_j that gives the maximum value to $P[h_j \mid \mathbf{t}_a, \mathbf{m}_a]$ is equivalent to finding h_j that achieves the maximum of

$\prod_{i=1}^{\ell} \hat{P}_{\text{model}}[t_i \mid x_i, h_j]$. Let $\mathsf{F}_{h_j}(x_i)$ denote the leakage for encryption block i with key h_j and the last byte of plaintext given by x_i. We model the measured side-channel leakage as a sum of a deterministic part and an independent random noise: $T_{x_i,h_c} = \mathsf{F}_{h_c}(x_i) + N_{\sigma^2_{\text{noise}}}$, where $N_{\sigma^2_{\text{noise}}}$ follows a Gaussian distribution with mean 0 and variance σ^2_{noise}. We have:

$$\prod_{i=1}^{\ell} \hat{P}_{\text{model}}[t_i \mid x_i, h_j] = \prod_{i=1}^{\ell} \frac{1}{\sqrt{2\pi}\sigma_{\text{noise}}} \exp\left(-\frac{1}{2}\left(\frac{t_i - \mathsf{F}_{h_j}(x_i)}{\sigma_{\text{noise}}}\right)^2\right).$$

Since $1/\sqrt{2\pi}\sigma_{\text{noise}}$ is positive and $\ln$ is an increasing function, we have:

$$\begin{aligned}
\text{argmax}_{h_j} P[h_j \mid \mathbf{t}_a, \mathbf{m}_a] &= \text{argmax}_{h_j} \prod_{i=1}^{\ell} \exp\left(-\frac{1}{2}\left(\frac{t_i - \mathsf{F}_{h_j}(x_i)}{\sigma_{\text{noise}}}\right)^2\right), \\
&= \text{argmax}_{h_j} \sum_{i=1}^{\ell} -\frac{1}{2}\left(\frac{t_i - \mathsf{F}_{h_j}(x_i)}{\sigma_{\text{noise}}}\right)^2, \\
&= \text{argmax}_{h_j} \sum_{i=1}^{\ell} -(t_i - \mathsf{F}_{h_j}(x_i))^2.
\end{aligned}$$

Define:

$$\mathsf{g}_{x_i,h_j}(t_i) := -\ell(t_i - \mathsf{F}_{h_j}(x_i))^2, \quad d_{h_j} := \frac{1}{\ell}\sum_{i=1}^{\ell} \mathsf{g}_{x_i,h_j}(t_i).$$

Then the distinguisher $\mathsf{d}((x_i, t_i)_i) := (d_h)_{h\in\mathcal{H}}$ is an additive distinguisher following the definition from [LPR+14]. We assume that the last byte of the plaintext in an encryption block follows a uniform distribution over integers from 0 to 255. Then by Corollary 2 of [LPR+14], d_h follows a Gaussian distribution with mean:

$$\mu_h := \mathsf{E}(d_h) = \frac{1}{\ell}\sum_{i=1}^{\ell} \mathsf{E}(\mathsf{g}_{x_i,h}(T_{x_i,h_c})) = -\sum_{i=1}^{\ell} \mathsf{E}((T_{x_i,h_c}\mathsf{F}_h(x_i) - \mathsf{F}_h(x_i))^2)$$

and variance:

$$\sigma_h^2 := \mathsf{Var}(d_h) = \frac{1}{\ell^2}\sum_{i=1}^{\ell} \mathsf{Var}(\mathsf{g}_{x_i,h}(T_{x_i,h_c})) = \sum_{i=1}^{\ell} \mathsf{Var}((T_{x_i,h_c}\mathsf{F}_h(x_i) - \mathsf{F}_h(x_i))^2).$$

Define $\mathsf{dif}_h(x_i) := \mathsf{F}_{h_c}(x_i) - \mathsf{F}_h(x_i)$, we have:

$$\mu_h = -\ell\sigma^2_{\text{noise}} - \sum_{i=1}^{\ell} \mathsf{E}(\mathsf{dif}_h(x_i)^2),$$

and:

$$\sigma_h^2 = 2\sigma^4_{\text{noise}}\ell + \sum_{i=1}^{\ell} \mathsf{Var}(\mathsf{dif}_h(x_i)^2) + 4\sigma^2_{\text{noise}}\mathsf{E}(\mathsf{dif}_h(x_i)^2).$$

For simplicity, we assume that the d_h are mutually independent. Then, $d_h - d_{h'}$ follows a Gaussian distribution with mean $\mu_h - \mu_{h'}$ and variance $\sigma_h^2 + \sigma_{h'}^2$. By [MOP08], Equation (4.29), the number of traces necessary to obtain $d_h - d_{h'} > 0$ with probability α is given by:

$$\frac{\sigma_h^2 + \sigma_{h'}^2}{(\mu_h - \mu_{h'})^2} q_\alpha^2,$$

where q_α is a quantile of the standard normal distribution, i.e., $P[X \le q_\alpha] = \alpha$, where $X \sim \mathcal{N}(0,1)$. Hence, to achieve success rate α, a lower bound on the number of traces needed is given by:

$$\min_h \max_{h'\neq h} \frac{\sigma_h^2 + \sigma_{h'}^2}{(\mu_h - \mu_{h'})^2} q_\alpha^2, \tag{5}$$

where the minimum is over a hypothesis h that has the last byte of l_0 and that of $\delta_0 \oplus k_0$ correct.