Revisiting Higher-order Computational Attacks against White-box Implementations

Houssem Maghrebi and Davide Alessio
UL Identity Management & Security, France

Keywords: White-box Cryptography, Masking, Higher-order Computational Attacks, Leakage Detection, AES.

Abstract: White-box cryptography was first introduced by Chow *et al.* in 2002 as a software technique for implementing cryptographic algorithms in a secure way that protects secret keys in an untrusted environment. Ever since, Chow *et al.*'s design has been subject to the well-known Differential Computation Analysis (DCA). To resist DCA, a natural approach that white-box designers investigated is to apply the common side-channel countermeasures such as masking. In this paper, we suggest applying the well-studied leakage detection methods to assess the security of masked white-box implementations. Then, we extend some well-known side-channel attacks (*i.e.* the bucketing computation analysis, the mutual information analysis, and the collision attack) to the higher-order case to defeat higher-order masked white-box implementations. To illustrate the effectiveness of these attacks, we perform a practical evaluation against a first-order masked white-box implementation. The obtained results have demonstrated the practicability of these attacks in a real-world scenario.

1 INTRODUCTION

1.1 White-box Implementations and Computational Attacks

In 2002, Chow *et al.* introduced the first white-box implementations of AES and DES block ciphers (Chow et al., 2003a; Chow et al., 2003b). The main idea behind was to embed the secret key in the implementation using a network of precomputed Look-Up Tables (LUTs) composed with some linear and non-linear random encodings to protect the intermediate states between the LUTs. The knowledge of one or all of these LUTs shall not give any information about the embedded secret key. To implement this design, two types of encodings shall be considered:

- Internal Encodings: are non-linear bijections applied to the input and/or the output of each LUT to hide its entries and/or its outputs. This category encompasses the so-called *mixing bijections* which are linear transformation applied to the input and output of each LUT to add more confusion to the implementation and ensure the cryptographic diffusion property.
- External Encodings: are bijective mappings applied to decode the plaintext from the sending process and to encode the resulting ciphertext to the receiving process.

To defeat white-box implementations, Bos *et al.* proposed in (Bos et al., 2016) the Differential Computational Analysis (DCA). This attack is the software adaptation of the well-known Differential Power Analysis (DPA) (Kocher et al., 1999). Specifically, the idea of the DCA consists in monitoring the memory addresses (as well as the stack, the CPU instructions, ...) accessed during the encryption process and recording them in the so-called *computation traces* (aka software execution traces). Then, a statistical analysis is performed to compute the correlation between a prediction of the targeted sensitive variable (that depends on a key guess) and each sample of the collected computation traces. The secret key corresponds to the key guess for which the highest correlation peak is obtained.

Since the publication of the DCA, several researchers have investigated the adaptation of either the well-studied side-channel attacks (Rivain and Wang, 2019) or the algebraic cryptanalysis techniques (Zeyad et al., 2019) to perform computational attacks in the white-box context. For instance, authors in (Rivain and Wang, 2019) proposed a software version of the collision attack and the Mutual Information Analysis (MIA). The experimental results performed on several publicly available white-

Maghrebi, H. and Alessio, D.
Revisiting Higher-order Computational Attacks against White-box Implementations.
DOI: 10.5220/0008874602650272
In *Proceedings of the 6th International Conference on Information Systems Security and Privacy (ICISSP 2020)*, pages 265-272
ISBN: 978-989-758-399-5; ISSN: 2184-4356

box AES implementations have shown significant improvements in terms of trace complexity compared to the DCA. Recently, Zeyad *et al.* have suggested the Bucketing Computation Analysis (BCA) in (Zeyad et al., 2019). This attack is inherently inspired by a cryptanalysis technique named statistical bucketing attack. The authors have demonstrated that this attack is very efficient to defeat some sophisticated white-box AES implementations (*e.g.* the WhibOx 2016 contest) with a fixed amount of traces (precisely 1024 traces to break a white-box AES implementation).

1.2 Masking and Higher-order Computational Attacks

Obviously, the well-studied side-channel countermeasures can be adapted and applied to protect white-box implementations. One common countermeasure is to apply masking which consists in sharing the intermediate variable into several mutually independent shares. In (Bogdanov et al., 2019), Bogdanov *et al.* investigated the approach of applying higher-order masking to resist DCA. Furthermore, the authors introduced, for the first time, the extension of the DCA to the higher-order case and analyzed the security of the masking countermeasure against these attacks in the context of white-box implementation.

1.3 Our Contributions

Following the investigations done in (Bogdanov et al., 2019), we propose in this work:

- **A Second-order BCA:** We extend in Sec. 3 the BCA to the second-order. Then, we demonstrate how this attack can defeat an internally encoded masked white-box implementation with exactly the same low trace complexity as for the first-order version studied in (Zeyad et al., 2019).
- **A Higher-order MIA:** We suggest in Sec. 4 two fashions of applying higher-order MIA in the context of white-box implementation. Both approaches are compared through practical experiments in terms of key-recovery efficiency and performance.
- **A Higher-order Collision Attack:** We study in Sec. 5 the higher-order version of collision attacks to defeat masked implementations. Then, we provide in Sec. 6 a comparison of the efficiency of the proposed attacks.

All our analyses are validated through practical experiments on the same first-order masked white-box AES implementation. Furthermore, we made the computation traces collected on this reference implementation publicly available (SM, 2019). The goal is twofold: (1) to ease the reproduction of our results by the white-box community and (2) to provide a commonly masked database (as no such traces from a masked implementation are available so far). In addition, the source code of some of our proposed attacks is publicly available as well.

2 PRELIMINARIES AND STUDY FRAMEWORK

2.1 Notations and Definitions

Along this paper, we use the following notations. The bold block capitals $\mathbf{X}$ denote matrices. The i^{th} column vector of a matrix $\mathbf{X}$ is denoted by $\mathbf{X}[i]$. The random variables are denoted by uppercase Latin letters, like X, while the lowercase letter x denotes a particular realization of X. The entropy $\mathbb{H}[X]$ of a random variable X aims at measuring the amount of information provided by an observation of X.

The intersection of two sets of values A and B is denoted by $A \cap B$ and is defined as $A \cap B = \{x \mid x \in A \text{ and } x \in B\}$. Two sets are disjoint if they have no elements in common, that is, A and B are disjoint if $A \cap B = \emptyset$.

To perform his attack, the adversary targets an intermediate sensitive variable which is a function of a plaintext x and a guessable secret key k^*. Then, for each key guess k, he computes a prediction of the target sensitive variable denoted $\phi(x,k)$ (or $\phi(k)$ for short) and measures the dependency between this prediction and the acquired traces. For our practical experiments, we target the Sbox output of the first round of an AES, *i.e.* $\phi(x,k) = \mathrm{Sbox}(x \oplus k)$.

2.2 Assumptions

The analyses and conclusions drawn in this work are done under the three following assumptions:

Assumption 1 (**Nibble-encoded Naked White-box Implementations**). *The targeted white-box implementations are nibble-encoded using an internal encoding. No external encoding is applied.*

Remark 1. *Nibble-encoding is the most common encoding size used for white-box AES implementations. We stress the fact that Assumption 1 is mandatory (according to (Zeyad et al., 2019)) for the study of the BCA described in Sec. 3. However, for the other investigated attack techniques in this work this assump-*

*tion can be relaxed (*i.e. *the results can be generalized to any encoding size).*

Assumption 2 (Similarity Encoding). *The most significant (respectively the least significant) nibbles of the masks used to protect the sensitive variable are encoded with exactly the same encoding function applied on the most significant (respectively the least significant) nibbles of the masked sensitive variable.*

Remark 2. *To the best of our knowledge, no masked white-box implementation is publicly available yet. We stress the fact that the publicly available Lee's CASE* 1 *implementation (Lee, 2018) partially involves masking techniques to only protect the first and last round of the implementation. Therefore, this implementation was broken when applying first-order DCA and BCA (Rivain and Wang, 2019; Zeyad et al., 2019)*[1]. *To apply masking to protect a full white-box implementation, several options could be investigated on how to internally encode the intermediate masked data and mask values. From our point of view, the most natural and straightforward way is to use the same encoding functions to protect these sensitive variables. Indeed, in (Lee et al., 2018) Lee* et al. *have followed the same strategy to apply their partial masking and they have justified their choice by the fact that it makes the unmasking operation easier. At the end, Assumption 2 is motivated by* (1) *the lack of reference masked implementations and* (2) *the fact that this encoding strategy is realistic (due to its simplicity). We keep the study of other encoding strategies in a masked context as a future work.*

Assumption 3 (Perfect Synchronization). *The targeted white-box implementations are only protected with higher-order masking. No shuffling or any random delays is introduced to de-synchronize the acquired computation traces.*

2.3 Targeted White-box Implementation

As introduced earlier, we study in this work the extension of some well-known side-channel attacks to the higher-order case when applied on masked white-box implementations. When a d^{th}-order masking is applied, each sensitive variable Z is split into $d+1$ shares $s_0, s_1, \ldots, s_d$ such that $s_0 \oplus s_1 \oplus \cdots \oplus s_d = Z$. Usually, the d shares $s_1, \ldots, s_d$ (called the masks) are randomly picked up and the last one s_0 (called the masked variable) is processed such that it satisfies the previous equality. Under assumption 3, we only focus on higher-order masking as a unique countermeasure applied to ensure protection.

For our experimental validation, we restrict the assessment of our proposed higher-order attacks against first-order masked white-box AES implementations. That is, we only evaluated the second-order versions. To do so, we implement a first-order masked white-box AES implementation under the three assumptions formulated in Sec. 2.2. This implementation is based on the classical Chow *et al.*'s white-box design relying on an internal encoding but with the major difference that every Tbox input and output is protected with an independent random mask. So, masking is applied on the whole implementation and all rounds. Moreover, our implementation is based on the well-known Global Look Up table (GLUT) method (Prouff and Rivain, 2007; Maghrebi et al., 2012). That is, the concatenation of the encoded masked data and the encoded mask is provided as an input to the masked Tboxes. The resulting size of our reference implementation is $41,5$ Megabytes.

Then, we collect the computation traces using an internal tool that monitors the read memory access during the execution of this first-order masked implementation. Each collected value during the acquisition phase is decomposed into several nibbles and then stored in the computation trace (Assumption 1). The collected traces are publicly available in (SM, 2019) to ease the reproducibility of our results. This trace database will serve as a reference for evaluating our proposed attacks in practice.

3 BUCKETING COMPUTATION ANALYSIS

3.1 Background

In (Zeyad et al., 2019), Zeyad *et al.* introduced a new computational attack called *Bucketing Computational Analysis* (BCA) to defeat unprotected white-box implementation. The core idea of BCA is that if two sensitive intermediate variables for two different plaintexts do not collide, then their encodings (using a deterministic bijection) should not collide as well. We recall in Algorithm 1 the pseudo-code describing the different steps of the BCA when applied on white-box AES implementations (Zeyad et al., 2019).

The BCA consists of 3 phases. During the pre-computation phase, for each key guess k, the attacker split the 256 plaintexts x (each corresponds to a different Sbox input) into two sets ($I_{0,k}$ and $I_{1,k}$) according to the resulting bucketing nibble $d = S'(x \oplus k) =$

[1]This implementation is not considered in this work for the aforementioned reasons.

Algorithm 1: BCA on white-box AES implementations.

Inputs: a targeted AES Sbox S of the first round and its corresponding S' (*s.t.* $\forall x \in GF(2^8)$, $S'(x) = S(x)\&0\mathrm{xF}$)
Output: good guess of the sub-key

$\ast\ast\ast$ Pre-computation phase $\ast\ast\ast$
Compute a set I of 256 plaintexts each corresponding to a different input of S
Pick two values d_0 and d_1 such that: $0 \leq d_0 < d_1 \leq 15$
for each key guess $k \in [0,255]$ **do**
    Split the plaintexts into two sets $I_{0,k}$ and $I_{1,k}$ w.r.t to the nibble d of S'
end for
$\ast\ast\ast$ Acquisition phase $\ast\ast\ast$
Acquire a set of 256 traces $\mathbf{T} = (t_{i,j})_{\substack{0\leq i\leq 255\\ 0\leq j\leq n}}$
$\ast\ast\ast$ Key-recovery phase $\ast\ast\ast$
Initialize a result vector R with 256 zeros
for each key guess $k \in [0,255]$ **do**
    Group the traces into $\mathbf{V_0}$ and $\mathbf{V_1}$ *w.r.t.* to the sorted plaintexts in $I_{0,k}$ and $I_{1,k}$
    for each sample j in the trace **do**
        if $\mathbf{V_0}[j] \cap \mathbf{V_1}[j] = \emptyset$ **then** : $R[k] = R[k]+1$
        end if
    end for
end for
The good sub-key guess corresponds to $k \in [0,255]$ that maximizes $R[k]$

$S(x \oplus k)\&0\mathrm{xF}$; *i.e.* x in $I_{0,k}$ (respectively in $I_{1,k}$) if $d = d_0$ (respectively if $d = d_1$). Regarding the choice of the values d_0 and d_1, the authors in (Zeyad et al., 2019) emphasized the use of $d_0 = 0$ and $d_1 = 0\mathrm{xF}$. In the sequel, we will use these two values as recommended. Then, during the acquisition phase, the attacker acquires 256 computation traces. Finally, the key recovery phase consists in sorting the computation traces into two *buckets* denoted $\mathbf{V_0}$ and $\mathbf{V_1}$ depending on the bucketing nibble d (whose value depend on a key guess). Then, it counts the number of disjoint columns in $\mathbf{V_0}$ and $\mathbf{V_1}$. The good key value corresponds to the key guess for which the number of disjoint columns is maximal.

Throughout several experiments, the authors have demonstrated that this attack is an efficient alternative to the DCA. Indeed, the required amount of traces to recover 4 bytes of the key is fixed (256 traces for a white-box AES implementation). However, as stated by the authors in (Zeyad et al., 2019), when masking is properly applied as a countermeasure, then the BCA fails to recover the key. Our goal is to extend the BCA to the second-order case to defeat masked white-box implementations. We keep the study of the generalization of BCA to an order greater than two as a future work.

3.2 Extension to the Second-order Case

Let's consider a first-order masked white-box AES implementation for which c computation traces of n samples each were acquired $\mathbf{T} = (t_{i,j})_{\substack{0\leq i\leq c\\ 0\leq j\leq n}}$. Then, according to Sec. 4 of (Bogdanov et al., 2019) and under Assumption 3 there exists a fixed couple (j_1^*, j_2^*) such that $(t_{i,j_1^*}, t_{i,j_2^*})$ are the shares (*i.e.* the mask and the masked value) of the target secret variable $S(x_i \oplus k^*)$ for any i in $[0,c]$. To check if the BCA can be extended to the second-order case, the arising question is whether $(\mathbf{V_0}[j_1^*] \oplus \mathbf{V_0}[j_2^*])$ and $(\mathbf{V_1}[j_1^*] \oplus \mathbf{V_1}[j_2^*])$ are disjoint sets only for the correct key guess k^*?

To answer this question, let's consider a couple of plaintexts (x_0, x_1) such that $x_0 \in I_{0,k^*}$ and $x_1 \in I_{1,k^*}$, B a 4-bit encoding function, and m_0 and m_1 the masks used during the encryption of x_0 and x_1 respectively. Then, on one hand, we have $x_0 \in I_{0,k^*}$ implies (under Assumption 2) that:

$$\begin{aligned}
&\mathbf{V_0}[x_0][j_1^*] \oplus \mathbf{V_0}[x_0][j_2^*] \\
&= B\Big(m_0\&0\mathrm{xF}\Big) \oplus B\Big((S(x_0 \oplus k^*) \oplus m_0)\&0\mathrm{xF}\Big) \\
&= B\Big(m_0\&0\mathrm{xF}\Big) \oplus B\Big(\underbrace{(S(x_0 \oplus k^*)\&0\mathrm{xF})}_{=0} \oplus (m_0\&0\mathrm{xF})\Big) \\
&= B\Big(m_0\&0\mathrm{xF}\Big) \oplus B\Big(m_0\&0\mathrm{xF}\Big) \\
&= 0 \ .
\end{aligned} \tag{1}$$

On the other hand, $x_1 \in I_{1,k^*}$ implies that:

$$\begin{aligned}
&\mathbf{V_1}[x_1][j_1^*] \oplus \mathbf{V_1}[x_1][j_2^*] \\
&= B\Big(m_1\&0\mathrm{xF}\Big) \oplus B\Big((S(x_1 \oplus k^*) \oplus m_1)\&0\mathrm{xF}\Big) \\
&= B\Big(m_1\&0\mathrm{xF}\Big) \oplus B\Big(\underbrace{(S(x_1 \oplus k^*)\&0\mathrm{xF})}_{=0\mathrm{xF}} \oplus (m_1\&0\mathrm{xF})\Big) \\
&= B\Big(m_1\&0\mathrm{xF}\Big) \oplus B\Big(0\mathrm{xF} \oplus (m_1\&0\mathrm{xF})\Big) \\
&= B\Big(m_1\&0\mathrm{xF}\Big) \oplus B\Big(\overline{m_1\&0\mathrm{xF}}\Big) \ .
\end{aligned} \tag{2}$$

Since B is bijective then $\mathbf{V_1}[x_1][j_1^*] \oplus \mathbf{V_1}[x_1][j_2^*]$ is non-null. Consequently, Eq. (1) and Eq. (2) prove that the sets $(\mathbf{V_0}[j_1^*] \oplus \mathbf{V_0}[j_2^*])$ and $(\mathbf{V_1}[j_1^*] \oplus \mathbf{V_1}[j_2^*])$ are disjoint for the good key guess. For a wrong key guess k, it is obvious that these sets have common values. Indeed, for any $x_0 \in I_{0,k}$ (respectively $x_1 \in I_{1,k}$) such that $k \neq k^*$, $S(x_0 \oplus k^*)$ (respectively $S(x_1 \oplus k^*)$) is a random value. Now, as the intersection between two sets containing random values is non-null (for some cardinality), then the sets $(\mathbf{V_0}[j_1^*] \oplus \mathbf{V_0}[j_2^*])$ and $(\mathbf{V_1}[j_1^*] \oplus \mathbf{V_1}[j_2^*])$ are only disjoint for the good key guess k^* which prove the soundness of our proposal. It is worth noting that the same soundness proof can be generalized to handle any value of the bucketing nibble pair (d_0, d_1).

So, the core idea of the second-order BCA is to search for two time samples (j_1^*, j_2^*) in the traces such that $(\mathbf{V_0}[j_1^*] \oplus \mathbf{V_0}[j_2^*])$ and $(\mathbf{V_1}[j_1^*] \oplus \mathbf{V_1}[j_2^*])$ are disjoints. To further discard the false positives, we emphases the use of the two following additional criteria that should be only fulfilled for the good key guess: (1) the set $(\mathbf{V_0}[j_1^*] \oplus \mathbf{V_0}[j_2^*])$ is null and (2) the set $(\mathbf{V_1}[j_1^*] \oplus \mathbf{V_1}[j_2^*])$ contains non-constant values (as the mask m_1 in Eq. (2) should be different from one encryption to another).

The complexity of the second-order BCA (as for any higher-order attack investigated in this work) highly depends on the size of the targeted area of interest to detect the leakages of the mask and the masked variable. To reduce this complexity, we recommend filtering the computation traces (*i.e.* removing the common constant values) as suggested in (Zeyad et al., 2019) and applying the different hints proposed in (Goubin et al., 2019) to reduce the dimensionnality of the traces.

Finally, we suggest an improvement for the first-order BCA reported in (Zeyad et al., 2019) to avoid the appearance of some false positive in some practical white-box evaluations. The idea is that, for each disjoint columns in $\mathbf{V_0}$ and $\mathbf{V_1}$, the attacker has to check if $\mathbf{V_0}$ and $\mathbf{V_1}$ are constant sets. Indeed, following the reasoning in Eq. (1) and Eq. (2), $\mathbf{V_0}$ and $\mathbf{V_1}$ should contain respectively $B(0)$ and $B(0\mathrm{xF})$ for the good key guess.

3.3 Experimental Results

To check the effectiveness of the extended BCA in a real-world scenario, we develop the second-order BCA. The source code of our implementation is publicly available in (SM, 2019). Then, we run the attack on the acquired traces of our masked white-box AES implementation described in Sec. 2.3.

The obtained results demonstrate that the 16 bytes of the AES key were recovered. It is worth noting that thanks to the new criteria suggested in this work, no false positives were detected for the false key guesses (*i.e.* $R[k]$ remains equal to zero when k is different from k^*).

4 MUTUAL INFORMATION ANALYSIS

4.1 Background

In 2008, Gierlichs *et al.* have proposed a new side-channel distinguisher called Mutual Information Analysis (MIA) (Gierlichs et al., 2008). It is an attractive alternative to the Correlation Power Analysis as it exploits any kind of dependency (linear or non-linear) between the leakage measurements and the predicted data. The MIA has been largely studied and tested on several implementations (Batina et al., 2011; Gierlichs et al., 2008; Prouff and Rivain, 2009). The core idea consists in estimating the mutual information between the leakage measurements L and the predictions $\phi(k)$ for every key guess k, that is: $\Delta_{\mathrm{MIA}}(k) = \mathbb{H}[L] - \mathbb{H}[L|\phi(k)]$. The correct guess of the key k^* corresponds to the key for which $\Delta_{\mathrm{MIA}}(k)$ is maximum. Since $\mathbb{H}[L]$ does not depend on the key guesses, then the adversary can equivalently look for the key that minimizes the conditional entropy $\mathbb{H}[L|\phi(k)]$. The major practical issue of mutual information is the estimation of the statistical distribution of the leakages. Several methods have been proposed in the literature: histograms, kernel density function, parametric estimation (Prouff and Rivain, 2009). We discuss in Sec. 4.2 how we dealt with this problematic to conduct our practical attacks.

To the best of our knowledge, the first report on the use of MIA in the context of white-box evaluation was provided by Rivain *et al.* in (Rivain and Wang, 2019). Indeed, the authors have assessed the publicly available white-box AES implementation NoSuchCon 2013) against an improved version of the MIA. The obtained results have proven that this attack is efficient to break internally-encoded implementation with only few traces (60 traces) compared to the DCA (4000 traces). This is expected as the dependency between the leakage and the predictions in the white-box context is non-linear (due to the usage of the internal-encoding to hide the implementation intermediate values). In the following section, our goal is to extend the MIA to higher-order context to target masked white-box implementations.

4.2 Extension to the Higher-order Case

In 2009, Prouff *et al.* have proposed in (Prouff and Rivain, 2009) a generalization of the MIA to higher-orders. Let's consider a d^{th}-order masked implementation and assume that the adversary knows exactly the manipulation times of the used masks and masked data. Hence, he is able to recover the corresponding

$(d+1)$-tuples of leakages $\mathbf{L} = (L_0, L_1, \ldots, L_d)$. Then, the higher-order MIA consists in finding the key guess that minimizes the conditional entropy $\mathbb{H}[\mathbf{L}|\phi(k)]$. It is noticeable that the probability density function (pdf) of the variable $\mathbf{L}|\phi(k)$ is often assumed to be a multivariate Gaussian mixture whose entropy can be estimated as for the first-order case using histograms, kernel density function or parametric estimation.

To efficiently apply the higher-order MIA (Prouff and Rivain, 2009) in the context of masked white-box implementation, we focus on the two following practical challenges:

The Choice of the PDF Estimation Method. In side-channel context, the estimation of the mutual information is a major practical issue as it involves some complex pdf estimation methods. It is worth noting that the results of these methods have a strong impact on the efficiency of the MIA (Batina et al., 2011). In white-box context, the computation traces contains non-noisy values. Hence, the estimated pdfs are discrete which makes the practical evaluation simpler (as also argued in (Rivain and Wang, 2019)). In such context, the histogram estimation seems to be the easier (natural) method to consider. That said, the optimal choice of the bin width is also an issue in statistical theory. For simple distribution, reasonable choices of the bin width are the Scott rule and the Freedman-Diaconis rule. However, under Assumption 1, the computation traces contain values in the range $[0, 15]$. Thus, the natural choice of the number of bins we consider is 16 (*i.e.* bin width equals to 1)[2].

The Choice of the Leakage Combination Function. To apply a d^{th}-order MIA in the white-box context, one can apply directly the side-channel approach described in (Prouff and Rivain, 2009). That is, the adversary has to consider d-tuples of leakages and look for the key guess that minimizes the conditional entropy of the multivariate pdf $(\mathbf{L}|\phi(k))$. Another approach, that we investigate in this work, consists in combining the d-tuples of leakages by applying the XOR function. Said differently, the idea is to minimize the conditional entropy of the uni-variate pdf $(\bigoplus_{i=0}^{i=d} L_i|\phi(k))$.

Indeed, under Assumption 3, the computation traces (and hence the leakages L_i) contains the exact value of the manipulated variable (*i.e.* the computation traces are noise-free). Thus, by applying the XOR combination, the adversary converts the leakages from multivariate to uni-variate context where the pdf estimation is much easier from a computational complexity perspective[3]. In the meantime, we could expect that both approaches should lead to the same key recovery efficiency. Indeed, since the computation traces are noise-free, then their processing using the XOR combination function will not induce an information loss (which is the case when performing higher-order DPA-like attacks that involve product and absolute difference combinations in a side-channel context where the traces are noisy (Prouff and Rivain, 2009)).

To verify this intuition, we compare in the following section both approaches when targeting our reference masked white-box AES implementation.

4.3 Experimental Results

We implement two versions of a second-order MIA: the first one is based on an estimation of a bi-variate pdf and the second one is based on the XOR combination of the leakages and then an estimation of the resulting uni-variate pdf. The pdf estimation is based on the histogram method with a fixed number of bins (16) as discussed previously. Then, we target our masked white-box implementation using 150 computation traces.

The obtained results have proven that both attack versions have succeeded to recover the complete AES key using. To compare the efficiency of both approaches, we compute the evolution of the success rate of each method according to an increasing number of used computation traces. The outcomes of this experiment have confirmed our claim that both methods have the same key recovery efficiency. In the meantime, it is worth noting that the XOR combination based second-order MIA is two times faster than the multivariate one. This behavior is quite expected for the reasons we discussed in the previous section. We provide the source code of our second-order MIA in (SM, 2019).

5 COLLISION ATTACK

5.1 Background

Recently, Rivain *et al.* have proposed a collision attack to defeat internally-encoded white-box implementations (Rivain and Wang, 2019). The proposed attack is inspired by the DCA. The major difference

[2]Commonly, the number of bins should be equal to $2^{(\text{encoding size})}$.

[3]It is obvious that the estimation of a multivariate pdf is more time-consuming than an uni-variate one (see for instance the discussions in (Prouff and Rivain, 2009; Gierlichs et al., 2010).

is that the collision attack is based on the processing of a pair of plaintexts (instead of one plaintext as for DCA) to build the prediction vector and the corresponding Collision Computation Traces (CCT). That is, for each key guess k and each pair of inputs (x_1, x_2), the adversary computes the Pearson correlation coefficient $\rho(\delta_{L(x_1)L(x_2)}, \delta_{\phi(x_1,k)\phi(x_2,k)})$ where δ_{xy} is the "vector adaptation" of the well-known Kronecker delta function and $L(x)$ is the computation trace collected while processing the input x. The secret key corresponds to the key guess for which the highest correlation peak is obtained.

So, the core idea of this collision attack is that if some sensitive variable collides for a pair of inputs, so does the corresponding encoded variable in the computation (as the encoding functions are bijective). One improvement of this attack we suggest from a performance perspective and which we validate through simulation and practical experiments is to consider the *equality distinguisher* instead of computing the correlation. Indeed, the adversary can count, for each key guess, the number of times the prediction vector $\delta_{\phi(x_1,k)\phi(x_2,k)}$ and the vector of the targeted samples in the CCT $\delta_{L(x_1)L(x_2)}$ are equal. This equality counter is maximum for the good key guess.

The authors in (Rivain and Wang, 2019) have successfully applied this attack on some publicly available white-box implementations and have demonstrated that the trace complexity is quite low compared to DCA. In the following section, we study the extension of this DCA-like collision attack to target higher-order masked white-box implementations.

5.2 Extension to the Higher-order Case

Let's assume that an adversary recovers the $(d+1)$-tuples of leakages $(L_0, \ldots, L_d)$ from a d^{th}-order masked white-box implementation. Then, to apply the $(d+1)^{\text{th}}$-order collision attack, he has to compute for each key guess and each pair of inputs (x_1, x_2) the following distinguisher:

$$\rho\Big(\bigoplus_{i=0}^{d} L_i(x_1) \oplus \bigoplus_{i=0}^{d} L_i(x_2), \delta_{\phi(x_1,k)\phi(x_2,k)}\Big) \quad . \qquad (3)$$

The soundness of our proposed d^{th}-order collision attack is inherently based on the soundness of the $(d+1)^{\text{th}}$-order DCA under Assumption 3. In fact, the $(d+1)^{\text{th}}$-order DCA is defined as $\rho(\bigoplus_{i=0}^{d} L_i(x), \phi(x,k))$ for every input x. When considering any pair of inputs (x_1, x_2), the $(d+1)^{\text{th}}$-order DCA rewrites $\rho(\bigoplus_{i=0}^{d} L_i(x_1) \oplus \bigoplus_{i=0}^{d} L_i(x_2), \phi(x_1,k) \oplus \phi(x_2,k))$. Now, since we are focusing on the study of the collision during the processing of two different inputs x_1 and x_2, the relevant values of the prediction vector are when $\phi(x_1,k) \oplus \phi(x_2,k) = 0$. Thus, one can transform the prediction vector from $\phi(x_1,k) \oplus \phi(x_2,k)$ to $\delta_{\phi(x_1,k)\phi(x_2,k)}$ to obtain the distinguisher described in Eq. (3). Said differently, the correlation is only computed for the pair of plaintexts for which a collision occurs for a key guess k (*i.e.* $\phi(x_1,k) = \phi(x_2,k)$).

Table 1: Comparison of the studied second-order attacks when targeting a masked white-box implementation.

Attack method	Execution time (s)	Number of traces
DCA	4.28	310
BCA	5.67	256
MIA (XOR)	15.58	150
MIA (multivariate)	34.11	150
Collision	70.15	500

5.3 Experimental Results

We validate first the soundness of the extended collision attack through simulation and we provide the source code of its second-order version along with the used simulated traces in (SM, 2019). Then, we run the attack on the computation traces collected from our reference masked white-box implementation. The obtained results have demonstrated the practicability of the attack. Indeed, we succeed to recover the AES key using 500 computation traces.

6 ATTACK COMPARISON

We provide in Tab. 1 a comparison of the results obtained for the studied attacks in this work. For the sake of comparison, we perform as well the second-order DCA as described in (Bogdanov et al., 2019). For a fair comparison, we target the same areas of interest to search for the leaking points of the mask and the masked sensitive variable. All the attacks are executed on a Linux machine with an Intel Core i7 processor at 3.60GHz and 16 GB of RAM.

The most efficient attack in terms of traces complexity is the MIA. This is expected due to its ability to capture the non-linear dependency between the leakage and the predictions. In terms of performance, the MIA based on the XOR combination is faster than the multivariate version. This could be explained, as already discussed in Sec. 4, by the fact that the estimation of the conditional entropy of multivariate random variables is more time-consuming compared to the univariate case. The second-order BCA offers the best trade-off between the execution time and trace complexity and hence it is a good alternative to the second-order MIA. Finally, when masking is involved then the DCA is better than the collision attack. Indeed, the collision attack (*w.r.t.* to Eq. (3)) can be seen as a particular case of the DCA where the correlation is only computed when a collision is detected. However, the collision attack remains a good candidate to consider in an unmasked context as demonstrated in (Rivain and Wang, 2019).

7 CONCLUSION

In this work, we considered the evaluation of higher-order masked white-box implementations. Indeed, we extended some well-known computational attacks to the higher-order context. The practical evaluation of these attacks had shown their efficiency to defeat masked white-box implementations.

As a future work, we intend to study these higher-order computational attacks when relaxing the assumptions formulated in Sec. 2.2.

REFERENCES

(2019). Supporting materials. https://github.com/Bucketing/HO-attacks-on-WB.

Batina, L., Gierlichs, B., Prouff, E., Rivain, M., Standaert, F.-X., and Veyrat-Charvillon, N. (2011). Mutual Information Analysis: a Comprehensive Study. *J. Cryptology*, 24(2):269–291.

Bogdanov, A., Rivain, M., Vejre, P. S., and Wang, J. (2019). Higher-order DCA against standard side-channel countermeasures. In *COSADE 2019, Darmstadt, Germany, April 3-5, 2019, Proceedings*, pages 118–141.

Bos, J. W., Hubain, C., Michiels, W., and Teuwen, P. (2016). Differential computation analysis: Hiding your white-box designs is not enough. In *CHES 2016*, pages 215–236, Berlin, Heidelberg. Springer Berlin Heidelberg.

Chow, S., Eisen, P., Johnson, H., and van Oorschot, P. C. (2003a). A white-box des implementation for drm applications. In Feigenbaum, J., editor, *Digital Rights Management*, pages 1–15, Berlin, Heidelberg. Springer Berlin Heidelberg.

Chow, S., Eisen, P. A., Johnson, H., and Oorschot, P. C. v. (2003b). White-box cryptography and an aes implementation. In *SAC 2002*, pages 250–270, London, UK, UK. Springer-Verlag.

Gierlichs, B., Batina, L., Preneel, B., and Verbauwhede, I. (2010). Revisiting higher-order dpa attacks:. In Pieprzyk, J., editor, *Topics in Cryptology - CT-RSA 2010*, pages 221–234, Berlin, Heidelberg. Springer Berlin Heidelberg.

Gierlichs, B., Batina, L., Tuyls, P., and Preneel, B. (2008). Mutual information analysis. In *CHES 2010*, volume 5154 of *Lecture Notes in Computer Science*, pages 426–442. Springer. Washington, D.C., USA.

Goubin, L., Paillier, P., Rivain, M., and Wang, J. (2019). How to reveal the secrets of an obscure white-box implementation. *Journal of Cryptographic Engineering*.

Kocher, P. C., Jaffe, J., and Jun, B. (1999). Differential power analysis. CRYPTO '99, pages 388–397, London, UK, UK. Springer-Verlag.

Lee, S. (2018). Lee's CASE 1 implementation. https://github.com/SideChannelMarvels/Deadpool /tree/master/wbs_aes_lee_case1.

Lee, S., Kim, T., and Kang, Y. (2018). A masked white-box cryptographic implementation for protecting against differential computation analysis. *IEEE Transactions on Information Forensics and Security*, 13(10):2602–2615.

Maghrebi, H., Prouff, E., Guilley, S., and Danger, J.-L. (2012). A first-order leak-free masking countermeasure. In Dunkelman, O., editor, *Topics in Cryptology – CT-RSA 2012*, pages 156–170, Berlin, Heidelberg. Springer Berlin Heidelberg.

Prouff, E. and Rivain, M. (2007). A generic method for secure sbox implementation. In Kim, S., Yung, M., and Lee, H.-W., editors, *Information Security Applications*, pages 227–244, Berlin, Heidelberg. Springer Berlin Heidelberg.

Prouff, E. and Rivain, M. (2009). Theoretical and Practical Aspects of Mutual Information Based Side Channel Analysis. In Springer, editor, *ACNS*, volume 5536 of *LNCS*, pages 499–518. Paris-Rocquencourt, France.

Rivain, M. and Wang, J. (2019). Analysis and improvement of differential computation attacks against internally-encoded white-box implementations. *IACR TCHES*, 2019(2):225–255.

Zeyad, M., Maghrebi, H., Alessio, D., and Batteux, B. (2019). Another look on bucketing attack to defeat white-box implementations. In Polian, I. and Stöttinger, M., editors, *COSADE*, pages 99–117, Cham. Springer International Publishing.