A Fast Characterization Method for Semi-invasive Fault Injection Attacks

Lichao Wu[1[0000−0002−7139−732X]], Gerard Ribera[2[0000−0001−8459−6283]], Noemie Beringuier-Boher[2[0000−0002−1312−3929]], and Stjepan Picek[1[0000−0001−7509−4337]]

[1] Delft University of Technology, The Netherlands
[2] Independent Researcher, The Netherlands

Abstract. Semi-invasive fault injection attacks are powerful techniques well-known by attackers and secure embedded system designers. When performing such attacks, the selection of the fault injection parameters is of utmost importance and usually based on the experience of the attacker. Surprisingly, there exists no formal and general approach to characterize the target behavior under attack. In this work, we present a novel methodology to perform a fast characterization of the fault injection impact on a target, depending on the possible attack parameters. We experimentally show our methodology to be a successful one when targeting different algorithms such as DES and AES encryption and then extend to the full characterization with the help of deep learning. Finally, we show how the characterization results are transferable between different targets.

Keywords: Physical attacks, Fault injection, Fast space characterization, Deep learning, Metrics

1 Introduction

A secure microcontroller or smartcard should be designed in such a way that no (or, as little as possible) secret information is leaked to the attacker and its integrity is protected. Still, there is an attack type that proved to be very powerful in the last decades and where, despite all the efforts, the attacker can obtain or modify the secret information. Such attacks are called implementation attacks as they do not target the algorithm's security but the weaknesses in its implementation. Two well-known types of implementation attacks are side-channel attacks (SCAs) and fault injection (FI) attacks. While those attacks are powerful, they can be also difficult to deploy due to a large number of choices one needs to make.

Semi-invasive attacks, a type of fault injection attacks, are widely used by attackers as well as during security evaluations in the industry due to their affordable and easy-to-repeat characteristics [1]. While semi-invasive attacks are powerful, they are not without limitations. First, the tuning of the parameters that play a role in the fault definition is a time-consuming and non-deterministic process. Using optical fault injection as an example, the required parameters to

perform evaluation are numerous: laser pulse amplitude, laser pulse width, spot size, delays (attack time interval), and scan locations. As a complete analysis considering all possible parameter combinations is not practical, the decisions involved in the process of the parameter selection are usually based on intuition and personal criteria of an attacker. Additionally, due to the differences between FI setups, the measurement results obtained from one setup cannot be easily reproduced by another. An attacker is consequently bound to repeatedly search for the optimal parameters in every attack scenario, for every sample, and setup. Finally, the existence of countermeasures on both hardware and software levels can further increase the difficulties in defining parameters such as delays and scan locations.

To solve these problems, a characterization of the target of evaluation (TOE) for the optimal parameter searching is necessary as the preliminary step of evaluation. Surprisingly, there is no formal approach for doing this. Manual testing on parameter combinations based on the attacker's experience is a common method to get an impression of the target behavior. Still, this approach is not able to provide good coverage of the impact analysis for all the parameter combinations when the investigation is time-constrained. For example, the combinations of a shorter laser pulse width and stronger laser pulse amplitude could be more effective in manipulating some short execution of the command such as integrity check; in contrast with the opposite parameter combinations, long execution, such as Flash writing, would be more easily interfered. Unfortunately, these optimal parameters cannot be covered by manual tests. Exhaustive search, on the other hand, can be a solution if a full characterization is needed but will require more time as a trade-off. Finally, techniques coming from the artificial intelligence domain could work well but face issues like the uncertainty of parameter selection. In terms of the parameter optimization, researchers explored techniques such as genetic [2, 3] and memetic algorithms [4] to improve the optimization approach. Although such approaches work well for voltage glitching or electromagnetic fault injection (EMFI), they are less universal for other fault injection approaches such as optical fault injection. More precisely, if the involved fault injection parameters are too strong, there is a high chance that the target will be damaged. Additionally, the obtained optimal parameters are limited to a certain fault injection setup as well as the sample under attack. Either the change of the setup or the sample will result in the change of the optimal parameters.

To speed-up the attack parameter identification while considering the coverage of the parameters, the development of strong and reproducible methodologies is of significant interest. Such methodologies should ensure a proper selection of the tested parameters and the effectiveness of an attack for various fault injection attack methods. Unfortunately, to the best of our knowledge, previous works only focused on optimizing the parameter selection for FI attacks. The methodology for the TOE characterization is still missing. Therefore, in this paper, we propose a methodology for the fast characterization of fault injection settings. The methodology is based on the construction of a sensitivity curve, which is

then used by the attacker for a proper selection of the fault injection parameters and their assessment. To that end, we propose two metrics, one to be used in the measurement phase and one in the evaluation phase. Next, we use deep learning for the full estimation of the characterization space based on a limited number of measurements. Finally, we show that the obtained characterization results can be transferred to different samples with the same target. Throughout the paper, we use optical fault injection to perform the attack because of its popularity and difficulty in terms of characterization. Nevertheless, our characterization method is compatible with other semi-invasive fault injection approaches.

In conclusion, the methodology we propose can boost the characterization process while keeping track of useful information. This can eventually lead to 1) a better estimation of the target behavior, 2) a proper selection of the fault injection settings, 3) a good reference when attacking different devices, and 4) an informative archive for future attacks.

1.1 Related Work

Fault injection is a well-researched topic already spanning a range of more than 20 years [5, 6]. Specifically, an optical fault injection attack is one of the most powerful attacks in this domain. Skorobogatov and Anderson introduced optical fault injection and attacked secure microcontrollers and smartcards [1]. There, the authors presented a countermeasure against such attacks (self-timed dual-rail circuit design technique) but concluded that such attacks are the most successful smartcard perturbation attacks as it is not easy to implement countermeasures. Although more advanced countermeasures have been developed in the later stage, optical FI attacks are still practical. S. Skorobogatov introduced a new optical fault attack type called fault masking attack [7]. Such attacks are aimed at disrupting the normal memory operation through preventing changes of the memory contents. Van Woudenberg et al. investigated optical fault injection on secure microcontrollers and concluded that the presence of countermeasures makes the attack more difficult but still possible [8, 9]. Note, while being very powerful, optical fault injection attacks are usually considered very complex due to the high costs of equipment and the preparation of the sample. More recently, Guillen et al. presented a low-cost fault injection setup capable of producing localized faults in modern 8-bit and 32-bit microcontrollers [10]. The authors showed how even such a low-cost setup can be used to successfully attack the Speck cipher.

When considering implementation attacks and artificial intelligence techniques, most of the work concentrated on side-channel analysis. There, machine learning and more recently deep learning techniques are playing an important role in profiling attacks that can outperform template attacks but also break implementations protected with countermeasures [11–13]. When considering fault injection, several works are investigating how to find fault injection parameters with evolutionary algorithms, but to the best of our knowledge, none of these works consider machine learning nor optical fault injection. Carpi et al. considered the usage of evolutionary algorithms to find the fault injection parameters

for supply voltage (VCC) glitching [2]. There, besides the evolutionary algorithms approach, the authors used three more search techniques. Next, Picek et al. extended this work by using a combination of an evolutionary algorithm and a local search to characterize the search space for voltage glitching as efficient as possible [4]. Maldini et al. used a genetic algorithm for finding fault injection parameters when considering electromagnetic fault injection (EMFI) [14]. There, the authors attacked the SHA-3 algorithm and reported 40 times more faulty measurements and 20 times more distinct fault measurements than by using a random search.

1.2 Our Contributions

In this paper, we consider semi-invasive fault injection attacks and fast characterization of the target behavior, which to the best of our knowledge, has not been explored before. More precisely, we introduce a methodology for semi-invasive fault injection that consists of:

1. New technique for searching for fault injection parameters consisting of a fast generation of a sensitivity curve and its evaluation, which is compatible with different FI techniques, attack scenarios, and TOEs.
2. Two metrics that enable us to properly guide the characterization and also assess it.
3. A novel approach based on deep learning classification that enables us to characterize the search space based on the limited number of actual measurements.

Besides these, from an attacker perspective, the use of the fast characterization method will significantly reduce the time needed to identify the optimal attack parameters. Additionally, because the characterization method increases the attack parameters coverage, the quality of the results will be improved and the chance of missing the optimal parameters will be reduced. To prove the efficiency of the proposed method, we provide detailed experimental results targeting the AES and DES ciphers implemented on a secured microcontroller. Finally, we then show that the characterization results are transferable towards different targets of the same type.

This paper is organized as follows. In Section 2, we discuss fault injection attacks, supervised machine learning, and neural networks. Next, in Section 3, we start by introducing our notation. Afterward, we present two new metrics we designed to help us better assess the performance of the attack and how to generate/evaluate the sensitivity curve. In Section 5, we discuss our experimental setup and results obtained after attacking samples with the AES and DES ciphers. Finally, in Section 6, we conclude the paper and present possible future research directions.

2 Preliminaries

In this section, we first describe the fault injection attacks, where we divide them into three types of attacks and discuss their major differences. We emphasize semi-invasive attacks due to their high-efficiency and low-cost properties. Subsequently, we briefly introduce the supervised learning paradigm, the general architecture of a neural network, and then broaden such a structure to the deep neural network. Finally, we discuss multilayer perceptron as the algorithm of choice in our experiments.

2.1 Fault Injection Attacks

Fault injection attacks aim at retrieving information or injecting faults to the target. Currently, many powerful techniques have been developed, all of which can be divided into three main categories - non-invasive, semi-invasive, and invasive attacks [15]. The main difference between the non-invasive and invasive attacks is in the approach of attacking the TOEs. To perform an invasive attack, it is required to remove at least part of the passivation layer to establish the contact between the probes and silicon [16]. Non-invasive attacks, on the other hand, mainly focus on investigating the settings that can be controlled externally [17], or passively measuring the running time, the cache behavior, the power consumption, and/or the electromagnetic radiation of the device through the package [18].

Semi-invasive attacks, standing in the middle of the two types of attacks discussed above, have their specific properties. Similar to the invasive attacks, they require direct access to the chip surface by removing the package, but the passivation layer is kept. A semi-invasive attack can be performed in a reasonably short time with much less expensive equipment than the invasive attacks. Finally, the skills and knowledge required to perform them also can be easily and quickly acquired [19]. From the approach perspective, semi-invasive attacks could be performed using a variety of tools such as IR light [20], X-rays [1] and other sources of ionizing radiation, electromagnetic fields [21], and body biasing [22].

2.2 Supervised Machine Learning

In the supervised learning paradigm, the goal is to learn a mapping f, such that $f : \mathcal{X} \to \mathcal{Y}$, given a training set of N pairs (x_i, y_i). Here, for each example x, there is a corresponding label y, where $y \in \mathcal{Y}$. This phase is commonly known as the training phase. The function f is an element of the space of all possible functions $\mathcal{F}$. Once the function f is obtained, the testing phase starts with the goal to predict the labels for new, previously unseen examples. In the case that Y takes values from a finite set (discrete labels), we conduct classification, while if the labels are continuous, we conduct regression.

2.3 Neural Networks and Deep Learning

A neural network is an interconnected assembly of simple processing elements, units or nodes, whose functionality is based on the biological process occurring in the brain [23]. In general, a neural network consists of three blocks: an input layer, one or more hidden layers, and an output layer, whose processing ability is represented by the strength (weight) of the inter-unit connections, learning from a set of training patterns from the input layer.

To improve computation ability, a standard approach is to add hidden layers to build a deep neural network. An example of the deep neural network is shown in Figure 1. With the help of multiple layers, a deep neural network can map complicated low-level details to high-level features progressively. Thus, deep neural networks can make a proper estimation of the output, where this adaption process is referred to as deep learning.

In this paper, we applied a commonly-used deep learning structure, multilayer perceptron (MLP) in our methodology. MLP is a feed-forward neural network mapping sets of inputs onto sets of appropriate outputs. It consists of multiple layers of nodes in a directed graph, with each layer fully connected to the next one. Each node in one layer connects with a certain weight w to every node in the following layer. The MLP architecture consists of at least three layers: one input layer, one output layer, and one hidden layer. Those layers must consist of non-linearly activating nodes [24].

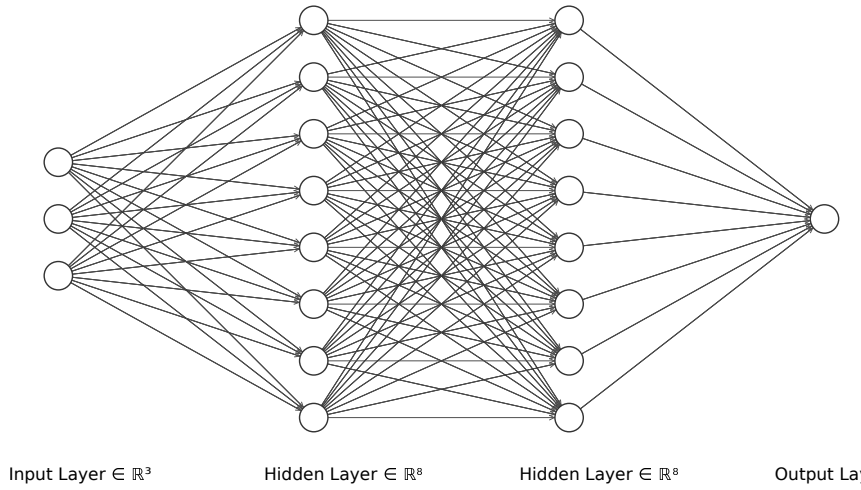


Fig. 1. An example of deep neural network with 2 hidden layers and 8 neurons per hidden layer (created with NN-SVG [25]). Note that it is enough to have more than one hidden layer to consider a certain architecture as deep learning.

3 Fast Characterization Methodology

A reliable characterization methodology can be used to obtain a quick impression of the influence caused to the target for a different combination of attack parameters. An attacker will use the outcome to better choose the settings to perform the attack in a later stage. However, there are several obstacles to build such a characterization methodology:

1. How to quantify the effect of the FI settings?
2. How to obtain a characterization of the impact that can be generated in a short amount of time?
3. How to map the behavior of the target to the characterization?
4. How to make sure that the characterization result is transferable between different targets?

The solutions to these problems are summarized with a work-flow presented in Figure 2. In general, one can observe that the attacker can divide his actions into two separate phases: 1) fast characterization of the target and 2) fault injection procedure. Our methodology concentrates on the fast characterization part as the fault injection procedure stems from it. To characterize the target in a fast and correct way, we first generate the sensitivity curve (described in Section 4). Next, we evaluate the measurements to further investigate the target behavior with different FI settings.

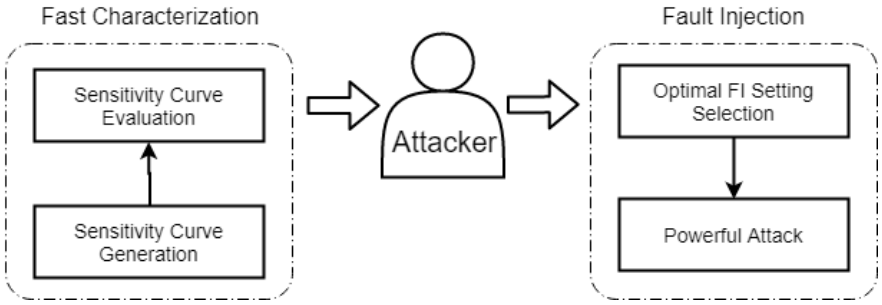


Fig. 2. An attack work-flow with proposed fast characterization methodology.

It should be noted that the attack location and time delay to inject the fault should be defined in advance, as they are initial conditions for the sensitivity curve generation. The attack location, for instance, can be inferred by reverse engineering techniques (i.e., IR-imaging) and a good understanding of the targeted fault model, while the Simple Power Analysis (SPA) can be used to define the attack time window. However, such analyses are out of the scope of this paper. Additionally, there are many other relevant parameters, such as the thickness of the silicon, that can influence the sensitivity of the target. However, it is a less interesting parameter in practice as it is difficult to control it precisely. In contrast, from an attacker perspective, the simplest and the most effective parameters to

work with are the parameters that can directly influence the strength of the injected fault, such as laser pulse width and laser pulse amplitude for optical fault injection. In this paper, we focus on characterizing these two parameters.

In this section, we start by introducing the notation used in this paper when discussing the behavior of targets. Next, we present two different metrics that enable us to better evaluate the performance of a fault injection process. One of the metrics (Level of Influence) measures the fault injection process and we use it in the proposed search algorithm while the other one (Impact Score) is used to evaluate the results of the fault injection. Note that throughout the paper, we use interchangeably the notions target, the target of evaluation, and its abbreviation TOE.

3.1 Notations

Fault injection attacks impact the behavior of the target, which can be noticed when its response to a target command deviates from the expected one. Those faulty responses can be used to categorize them into verdict classes that correspond to the effectiveness of the measurement (i.e., attack attempt). The possible classes for each measurement are listed in the ascending order based on their relevance for the attacker.

1. NORMAL: TOE behaves as expected.
2. RESET: The attack is detected and TOE resets.
3. MUTE: TOE stops communication. This type of response can be caused either by hard failures caused by the attack (i.e., the chip doesn't work anymore) or can be the response when the attack is detected.
4. CHANGING: TOE fails to detect the injected faults and returns unexpected values.
5. SUCCESS: TOE fails to detect the injected faults and returns abnormal but exploitable values.

Note that an exploitable fault is a fault that can be used to obtain more critical information (e.g., retrieve encryption key with Differential Fault Analysis (DFA) [26]) or perform additional malicious activities (e.g., install unauthorized software). In this paper we attack two popular encryption algorithms: AES and DES. Specifically, we define an exploitable fault as faulty cipher outputs, as the encryption key can be retrieved with DFA with these outputs. A non-exploitable fault, on the other hand, can be any other outputs, such as status word or unrelated data stored in other addresses. It also worth to mention that when attacking a device with fault injection, different types of unexpected results can be outputted and are difficult to classify. The situation becomes even worse when targeting different types of devices as the implementations are also different. To simplify the characterization and to abstract from the underlying fault model, we classify the faults on the algorithmic level instead of on the hardware level.

In this paper, the optical FI technique is used for the experiments. The main attack parameters - the laser voltage (energy) and laser pulse width are denoted with upper-case letters X and Y, while their realizations are given in the lower-case letters x and y. More precisely, the search boundaries for these two FI

settings are X_{min}/X_{max} and Y_{min}/Y_{max}. The search steps are represented by X_{step} and Y_{step}.

3.2 Metrics Definition

Level of Influence The Level of Influence (LOI) represents the percentage of responses that are different from the expected (NORMAL response) in the total number of attempts, which can be used to quantify the impact of the attack parameter set. For instance, by decreasing the laser pulse amplitude or the duration, the fault injection is less effective and the target tends to behave normally, thus having a low influence. In contrast, by increasing these settings, there is a higher possibility that the target is influenced by the attack, which will eventually increase its influence on the target behavior. The LOI metric can be calculated as follows:

$$LOI = 1 - \frac{Quantity_{normal}}{\sum^{class} Quantity_{class}}. \tag{1}$$

Here, $Quantity_{normal}$ represents the number of NORMAL responses while $Quantity_{class}$ represents the number of the specific class occurrences during the whole measurement process. The LOI directly links to the level of interference introduced by the injected fault to the targets.

Impact Score The outputs of the TOE under fault injection are divided into several classes (see Section 3.1). To further clarify the effect of each FI settings and to optimize the parameter selection in the later attack phase, we assign weights to each class based on its significance and eventually come up with a score based on every measurement result. As this score directly reflects the effects of the FI with respect to the target behavior, we denote this metric Impact Score (IS).

The Impact Score metric aims to show the relevance of the measurements that are acquired during the generation of the sensitivity curve (see Section 4). By assigning different weights to the different classes obtained, an attacker can identify if some of the parts of the curve are more relevant and could potentially lead to a successful manipulation.

In practice, class SUCCESS has the highest priority of all the classes and is assigned the largest weight. Differing, the class NORMAL (indicating the target behaves normally) is linked to a small weight. The IS metric can be calculated as:

$$IS = \frac{\sum^{class} Quantity_{class} \cdot Weight_{class}}{\sum^{class} Quantity_{class}}, \tag{2}$$

where $Weight_{class}$ represents the assigned weight for a corresponding class. In the experiments presented in this paper, the classes SUCCESS, CHANGING, MUTE, RESET, and NORMAL have weights 20, 10, 2, 0.5, and 0, respectively.

The weights are adjusted based on the experience of the attacker and the rationale behind is defined after an assessment of the hypothetical fault model that leads to such responses.

4 Sensitivity Curve

In this section, we start by introducing the concept of the sensitivity curve. Afterward, we discuss how to generate such a curve by first finding the "golden" point and then applying the sensitivity curve search algorithm. Finally, we discuss how to evaluate the sensitivity curve through Impact Score or deep learning classification process.

4.1 Setting

To obtain a characterization algorithm that has a good parameter coverage, is less time-consuming, and is universal for different scenarios, several methods from simple (e.g., exhaustive search with large scan step, binary search) to complicated (e.g., genetic algorithm, deep learning) have been tested. The comparison of different architectures is not shown due to the lack of space and redundancy in obtained results. We observed that simple algorithms are predictable which is ideal for the TOE characterization but normally less time-efficient. In contrast, complicated approaches tend to rely on the number and quality of the obtained data. However, these algorithms work unstable as the number of data sets we obtained is extremely limited. In the worst case, a non-converged model can lead to the target being damaged by the undesired parameter selection.

Therefore, the ideal algorithm for the characterization should stand in the middle of these two extremes. In other words, it should be deterministic, but not highly data-dependent. Fortunately, the sensitivity curve, which consists of a set of FI settings that cause a similar impact on the TOE, perfectly fulfills our requirements.

Three sensitivity curves with different LOIs are given in Figure 3; each point on the curve has a similar impact on the TOE behavior. There, with sensitivity curves, one can estimate that the injected fault (X and Y axes represent the FI settings) can be ignored at the left side of the curve with 5% LOI; while the target will behave almost always abnormally at the right side of the curve with 90% LOI. Moreover, the figure presents multiple possible selections of the fault injection settings that can lead to the same LOI. For instance, to achieve 50% of the LOI, besides choosing the parameters in the middle of the curve, an attacker can achieve a similar result by selecting smaller x and larger y or vice versa. It is possible that the sensitivity curve is not decreasing monotonically as shown in Figure 3. Nevertheless, the sensitivity curves act as contour lines in the parametric coordinate system, which can be used to estimate the quantity of impact with different FI settings. Furthermore, the presence of the sensitivity curves provides the attacker with a multiple choice in setting selection: although the LOI is the same, appropriate selection of the FI settings based on the attack

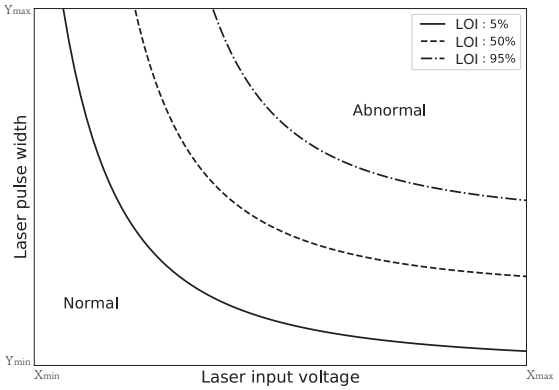


Fig. 3. An example of the sensitivity curves with different LOIs. From here, the normal and abnormal behaviour of the target can be estimated.

scenarios may lead to a more powerful attack. Therefore, we use the sensitivity curve for TOE characterization.

To conclude, if compared with other approaches, the advantages of the sensitivity curve-based characterization are the followings:

1. The sensitivity curve defines the natural boundary between the "weak" and "strong" FI settings, which present a rough overview of the target behavior.
2. The input of the sensitivity curve delimits the number of the parameter combinations to be examined, thus it is more time-efficient.
3. Since the LOI of a sensitivity curve is defined by an attacker, it resolves the problem of an FI setting selection through a genetic algorithm or random search.
4. The proposed methodology can be applied to other semi-invasive FI methods that follow the assumption that the strength of the setting is positively correlated to the level of impact on the target, such as EMFI and Body Biasing Injection (BBI).

4.2 Sensitivity Curve Generation

In general, the searching of the sensitivity curve relies on iterative performing of measurements and calculating the statistics to decide the next setting to be tested until the end condition is fulfilled. The statistics (LOI) that are calculated are based on the types of output recorded in each setting combinations. To make a clear description, the search algorithm is split into two phases: first, determine the "golden point" and then search for the entire curve.

Finding the "Golden Point" The golden point (X_{golden}, Y_{golden}) represents the first obtained FI setting that with target LOI defined by an attacker (C_{target})

and acts as the reference for the curve searching in the later step. To find such a point, we use the diagonal search algorithm. The diagonal search algorithm is performed by increasing the values of the FI parameters simultaneously with a fixed step as shown in Figure 4. Note how the search progresses in a number of steps (in our example, 6) before reaching a point on the sensitivity curve. The diagonal search algorithm ensures to start testing with weak laser settings and then gradually going stronger. Indeed, some approaches may lead to faster convergence. However, during the experiments, we noticed that the chip sensitivity towards the laser can vary dramatically between targets (i.e., different types of microcontrollers). In other words, a laser setting that does not have any influence on one product may destroy another product immediately. Consequently, the diagonal search algorithm is selected to ensure the tested product being alive throughout the characterization process as well as to broaden the usage of our methodology towards different products.

It is worth to note that the diagonal search cannot always guarantee to find the FI settings with exact C_{target} value. In many cases, the LOI can exceed the target when performing the search. Therefore, we introduce the $C_{tolerance}$ to broaden the range search of the golden point: if the LOI of the tested FI setting is within the range of $C_{target} \pm C_{tolerance}$, the applied FI setting can be counted as the golden point. In cases when the current LOI exceeds the maximum range (C_{target}+$C_{tolerance}$) but no golden point is observed, a binary search is performed to trace back to lower settings and search for the golden point within the range of tolerance.

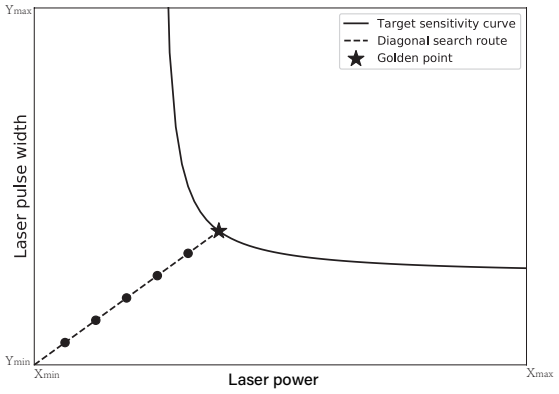


Fig. 4. A depiction of the diagonal search. The golden point represents the first obtained FI setting with the target LOI.

Curve Searching Once the golden point is obtained from the diagonal search, the search for the sensitivity curve can be executed. As discussed in Section 4.2,

the golden point is obtained in a diagonal route, but there are still areas on its left and right-hand side to be characterized. Therefore, to localize the sensitivity curve in the whole parameter plane, the curve search is performed in both directions individually, while they start with the golden point. As the search strategies for both directions are the same, the search algorithm to the left (X_{min}) direction is given in Algorithm 1. Curve search on the right-hand side can be realized by adjusting the *while* condition as well as the x increment step.

Algorithm 1 Sensitivity curve search.

```
1: function SEARCHING_LEFT(X_golden, Y_golden, C_target, C_tolerance)
2:     data ← []
3:     x ← X_golden
4:     y ← Y_golden                                  ▷ Initialize (x, y)
5:     while x − X_step > X_min do                   ▷ Search from the left plane
6:         x ← X_prev − X_step
7:         LOI ← DoTest(x, y)                        ▷ Test with setting (x, y)
8:         if LOI < C_target − C_tolerance then
9:             y ←BinarySearch(y, Y_max)             ▷ Search with stronger settings
10:        else if LOI > C_target + C_tolerance then
11:            y ←BinarySearch(y, Y_min)             ▷ Search with weaker settings
12:        data ← data + [x, y, LOI]
13:    return data                                   ▷ Return all of the tested data
```

The function $DoTest(x, y)$ performs a measurement with a combination of the FI setting x and y. $BinarySearch(a, b)$ represents the binary search in the range from a to b. The main idea of Algorithm 1 is to first iteratively obtain the measurements and second, calculate the statistics to decide the next pairs of settings. Specifically, by varying x while keeping the y obtained by the previous steps, the algorithm can keep track of the changing tendency of the target sensitivity curve. Moreover, the usage of the parameters from the previous test delimits the range for the binary search, thus accelerating the whole characterization procedure.

Instead of using a fixed value, X_{step} should be adjustable for different conditions. For instance, increasing X_{step} to accelerate the characterization when the slope of the sensitivity curve is close to zero while reducing its value to evaluate more FI settings when the slope becomes steep. To realize this functionality, a new variable Y_{diff}, which stands for the value difference between the current y and the previous y (Y_{prev}), is added to the algorithm. The pseudocode of the step adjustment function is shown in Algorithm 2.

4.3 Sensitivity Curve Evaluation

The sensitivity curve provides the attacker with a quick impression of the target behavior (through the LOI metric) with different FI settings. To further benefit

Algorithm 2 Step adjustment

```
1: function ADJUST_XSTEP(X_step, Y_step, Y_prev, y)
2:     Y_diff ← absolute(Y_prev − y)
3:     if Y_diff <= Y_step then
4:         return X_step * 2
5:     else
6:         return X_step / 2
```

from the performed measurements, the attacker can use techniques to visualize the data differently with the Impact Score metric and to obtain an overview of the different setting relevance in FI. Additionally, he can even estimate the non-measured parameter combinations with a machine learning algorithm.

Impact Score Evaluation As described in Section 4.2, the generation of the sensitivity curve is based on searching the FI settings with a similar LOI. Although the target behavior can be estimated based on the curve, it is difficult to define the optimal parameters which can lead to more significant responses. Indeed, LOI only distinguishes between NORMAL and non-NORMAL responses. To fully evaluate the performance of one setting, the non-NORMAL response should be additionally classified based on its significance.

Taking advantage of its wide setting selection, the sensitivity curve is a good candidate for evaluating the effectiveness of the FI. Therefore, the curve is regenerated with the IS metric to obtain the optimal setting for fault injection. Specifically, by calculating IS for each parameter combination, the relevance of the measurement can be quantified: a larger Impact Score represents the existence of higher-priority responses, indicating that the corresponding setting is more preferable for the later attacks.

Impact Estimation with MLP In practice, the assessment of attacking the non-measured area is a part of the evaluation and comes from the attacker's decision. Various advanced techniques can be used to help the attacker to estimate the impact in the non-measured areas. Here, function regression, realized by MLP with gradient descent, is used to build the relationship between its input (FI parameters) and output (LOI). A converged model can provide a proper estimation of the impact that can be caused in the target with different parameters.

However, the prediction accuracy highly relies on the training data. Indeed, the sensitivity curve provides several unique data sets, but the prediction of the untested locations is still challenging, as the number of the training sets is extremely limited while we aim at predicting huge amounts of parameter combinations in a wide range. In this paper, multilayer perceptron is selected for prediction. Compared with other machine learning structures and statistic methods, MLP dramatically reduced the prediction error especially in the excessive area from weak to the strong parameter (the region an attacker cares

about most) with the help of the deeper layers. Although higher precision of the prediction can be obtained by using more data (e.g., by generating another sensitivity curve with different LOI), MLP seems to be the best solution to provide an overall estimation of the target behavior without additional tests (costs). Moreover, in our case, MLP is less sensitive to the distribution/number variation of the training sets and can always extract features from a limited amount of data and thus can improve the robustness of our methodology.

The cross-entropy is implemented as the loss function to classify the discrete data from the sensitivity curve. By minimizing the loss function during iterations, the MLP can estimate the LOI with different FI settings, whose accuracy is further evaluated by calculating the offset between the predicted and measured data. Note that we consider the prediction result as reasonable if the prediction error is small when compared with the test data and the plots fit the shape of the sensitivity curve. Although the sample's behavior under attack can vary from the prediction due to the prediction error and many other reasons, the presented prediction methodology can provide an attacker with a proper estimation of the overall sample behavior, which leads to a better selection of the parameters.

5 Results

In this section, we start by introducing our experimental setup. Then, we present the results obtained for DES and AES settings using the presented fast characterization methodology. Finally, we validate the transferability of the characterization result by repeating the characterization for a different sample of the same TOE.

5.1 Experimental Setup

In all our experiments, we use a TOE based on a high-performance 32-bit microcontroller realized in Complementary Metal Oxide Semiconductor (CMOS) technology with 4MHz clock frequency. Due to confidentiality reasons, we are not able to disclose the details of the targets. Still, we are confident to note that the proposed method is compatible with various types of devices, as it was proved to be efficient with multiple devices that are not listed in the paper due to the page limit.

No FI specific countermeasures are implemented at the hardware level. For the experimental purpose, we present two different attack scenarios on software implementation of cryptographic algorithms, one targeting the beginning of the last round of Data Encryption Standard (DES) cipher and another one targeting the beginning of the last round of Advanced Encryption Standard (AES) cipher. Note that we used Single Power Analysis (SPA) to identify the encryption rounds. In both cases, we present a fast characterization that could be used by an attacker to perform the attack in a later stage to obtain faulty ciphers that can be used to run a DFA attack [27].

Experiments shown in this paper are performed on the Flash decoders as we assumed they are the most vulnerable part for light manipulation. The attack locations are uniformly distributed on the entire scan area. The FI setup used to perform the measurements is an optical fault injection setup using an IR light (1 024 nm) long-pulse laser which is one of the most powerful solutions for an optical fault injection attack. Since this light source is less effective when attacking the front-side of the sample as it cannot penetrate through the metal layers, we concentrated on attacking the backside (silicon side). To fully demonstrate as well to characterize the chip behavior with different laser settings, we selected a wide range of parameters that are used during the searching algorithm. The details are given in Table 1 while the MLP hyper-parameters for the LOI prediction are in Table 2.

Table 1. Parameters for the search algorithm.

Parameter	Value
Laser Pulse Width	[1, 50] µs in a step of 1 µs
Laser Voltage (Pulse Amplitude)	[0.05, 0.6]V in a step of 0.01 V
Target LOI	0.5
Searching Tolerance	0.05

Table 2. MLP hyper-parameters.

Parameter	Value
Architecture	[2, 8, 6, 6, 5, 1]
Activation	4 ReLU + 1 Sigmoid
Learning Rate (α)	0.2
Decay Rate	α * 0.97 per 1 000 epochs
Regularization	L2
Iterations	50 000

5.2 Characterization for the DES Encryption Attack

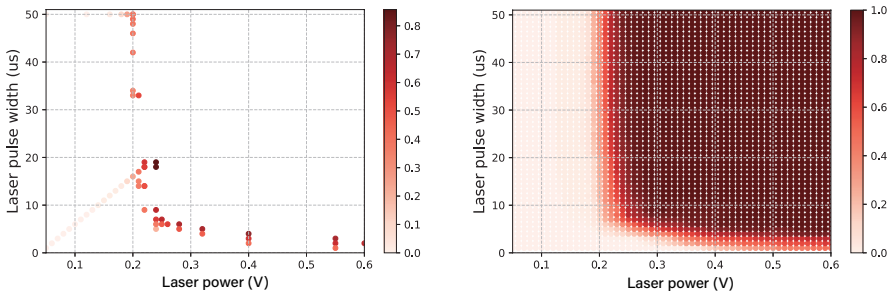
The DES encryption process is the target execution in this attack scenario. The attack time interval is delimited with SPA (Simple Power Analysis). The fast characterization is launched to assess the FI settings that might potentially lead to a successful attack (i.e., faulty ciphertexts).

Three steps are performed during the characterization procedure: first, generating the sensitivity curve, followed by the impact estimation using a deep learning algorithm, and finally evaluating the curve with the IS metric. During the first step, all the measurements are acquired. The second and third steps

belong to the evaluation phase. The generation of the sensitivity curve and the impact estimation using deep learning are based on the LOI metric while the third step is based on the IS metric.

Level of Influence for DES The characterization result based on the proposed algorithm is depicted in Figure 5a. For comparison purposes, a full-characterization was performed and the LOI graph of an exhaustive scan with a full range of settings is shown in Figure 5b. The color of the dots represents the value of the LOI metric. The test run of Algorithm 1 to perform the fast characterization (59 measurement points) was obtained within 2 hours while the full-characterization (3 080 measurement points) took more than a week of measurement time.

As a remark, each training datum represents the results from different attack locations. Attacking more locations can better represent the sample's behavior with certain laser parameters, but will spend more time as a trade-off. Here, we performed an exhaustive scan with more than 3 000 tests for the validation purpose, where due to the time constraints, we have to control the cost of the training data in an acceptable range (around 4 minutes per test).



(a) LOI: characterization. **(b)** LOI: exhaustive scan.

Fig. 5. LOI distribution with different fault injection settings.

From the result, the outline of the sensitivity curve, which acts as the boundary between "week" and "strong" parameters, can be estimated with the measured data. Based on this curve, the impact of the target on different FI settings can be estimated. Besides that, additional information can be extracted from the graph:

1. FI becomes effective when the laser voltage is larger than 0.2 V.
2. Similar LOI can be achieved with completely different setting combinations.
3. Laser voltage is more influential in FI than the laser pulse width.

The usage of this information depends on the attack scenario. For example, if the attack scenario is to skip an instruction execution, short pulses might be preferred; whereas to corrupt a memory write (long operation), longer pulses

could be more appropriate. Nevertheless, an attacker can benefit from these inputs in the next phase of the attack.

The MLP (as described in Table 2) is used to predict the LOI with all FI setting combinations, trained by the data obtained during the characterization process. In this attack scenario, 59 training set pairs, with two FI settings as features and Level of Interest values as labels, are collected from the sensitivity curve. The plot of the loss with respect to the epoch numbers during the training is shown in Figure 6.

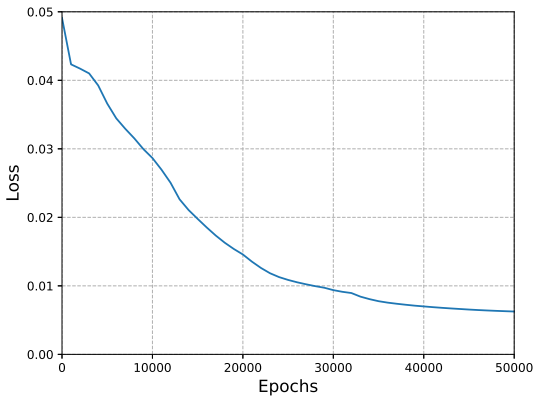
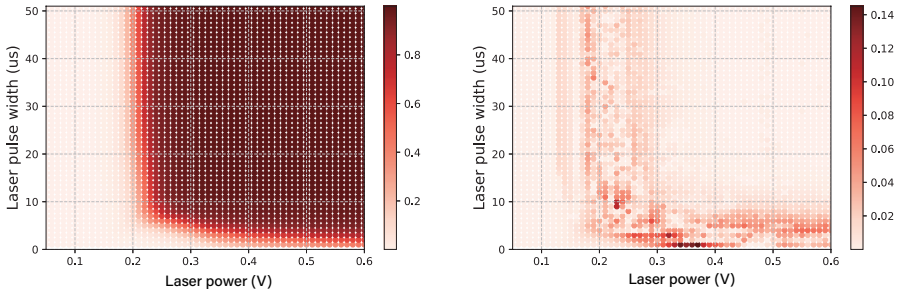


Fig. 6. LOI prediction for DES: loss versus epoch numbers.

As shown in Figure 7a, the prediction result matches the measured data with the majority of the setting combinations. The prediction error plotted in Figure 7b is also close to the sensitivity curve: the maximum error is 0.14 and the average error is 0.009. The prediction results indicate the capability of deep learning in predicting LOI with a limited number of training sets, which offers a proper estimation of the target behavior in significantly less time than a full characterization.

Impact Score for DES To further investigate target behavior, Impact Scores are calculated (Figure 8a) based on the measurements performed during the generation of the sensitivity curve. The IS results from the exhaustive scan are shown as the reference (Figure 8b).

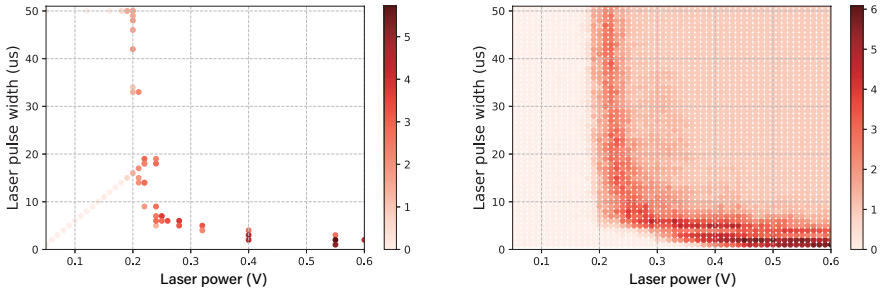
From Figure 8a, a higher IS can be obtained with shorter laser pulse width but stronger laser voltage, indicating the high probability in obtaining more significant output in this region. Indeed, this assumption can be proved by Figure 8b with IS for all setting combinations. Since the IS-based sensitivity curve only covers a few of the setting combinations, other, untested optimal settings could still exist. Still, this curve provides a general layout for the settings with



(a) Prediction result using a five-layer neural network.

(b) Error plot when comparing with the full-characterization measured data.

Fig. 7. Prediction result with a deep neural network.



(a) IS: characterization.

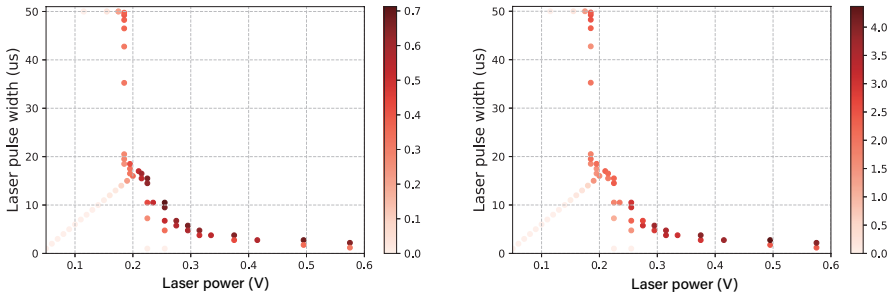
(b) IS: exhaustive scan.

Fig. 8. IS distribution with different fault injection settings.

better relevance from the measurements performed, which can eventually lead to a better parameter selection for a later attack stage.

Transferability of the DES Characterization Results In general, two factors are influencing the characterization result: sample's behavior under attack and the setup used for the attack. Any variation of these two factors will make the characterization result less usable. In terms of transferability of the characterized parameters, since we use the same type of TOE and attack different samples with the same setup, the resulting parameters should be transferable (indeed, the impact of process variations should be negligible for optical FI). To prove this assumption, we generated the sensitivity curve with the LOI and IS metrics on a different sample. The results are shown in Figure 9.

In terms of LOI, besides some small differences due to the variation of the chip alignment and laser focus, the result is similar when compared with Figure 5a. The IS, on the other hand, also shows its consistency when comparing with Figure 8a, as it also indicates that the shorter laser pulse width with stronger laser



(a) LOI: characterization with a different sample.

(b) IS: characterization with a different sample.

Fig. 9. Characterization results with a different sample targeting DES encryption.

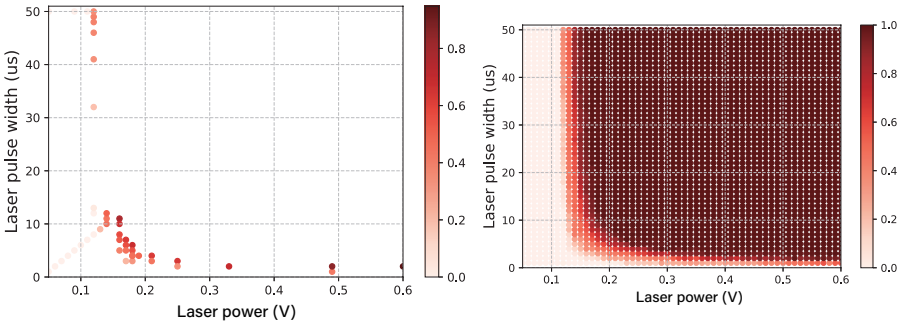
voltage can lead to higher impact scores at the same parameter range. Therefore, since the shape of the curve, LOI, and the corresponding IS tested on two different samples match with each other, we conclude that the characterization result from one sample is transferable to a different sample of the same TOE.

5.3 Characterization for the AES Encryption Attack

To verify the proposed methodology in different conditions, we performed an additional FI experiment with another laser setup of the same type. This experiment aims to manipulate the encryption of AES software implementation. Similar to the previous experiment, SPA techniques are used to delimit the attack time interval. The building block to be targeted is kept the same (Flash decoders).

Level of Influence for AES As for the DES cipher, a characterization was performed to obtain a LOI graph. The sensitivity curve is shown in Figure 10a (47 measurements) while its full-characterization counterpart is presented in Figure 10b (3 800 measurements). When comparing this characterization result with the one targeting the DES encryption (Figure 5a), we can observe the differences in setting selections for comparable LOIs. This difference can be due either to the use of a different laser setup or to the different attack scenarios.

Once the LOI graph was obtained, the same MLP architecture was used to map the LOI with all the FI setting inputs from the data measured during the sensitivity curve generation. Again, we plot the loss with respect to the epoch numbers during the training. The result is shown in Figure 11. By comparing the prediction results (Figure 12a) with the full-characterization (Figure 10b), we can confirm that the LOI tendency is properly estimated. To evaluate the prediction error, the difference between the two is plotted in Figure 12b. Although the



(a) LOI: characterization. **(b)** LOI: exhaustive scan.

Fig. 10. LOI distribution with different fault injection settings.

error can be further delimited by tuning the hyper-parameters of the network architecture or increasing the number of measurements during the sensitivity curve generation, the effectiveness of the MLP for LOI estimation is verified.

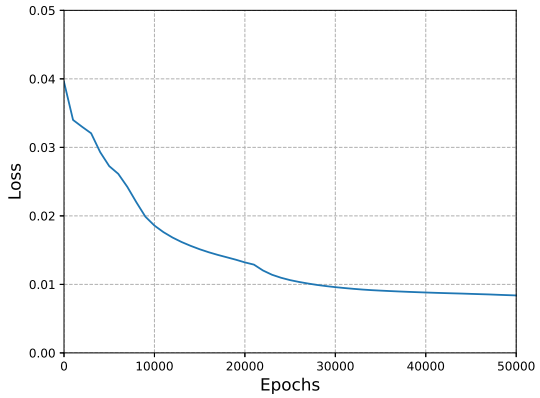
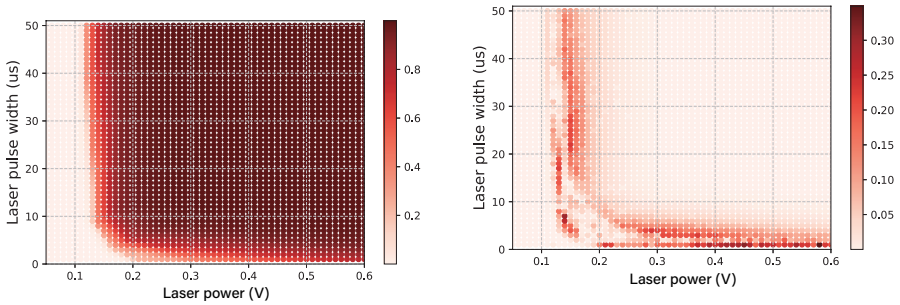


Fig. 11. LOI prediction for AES: loss versus epoch numbers.

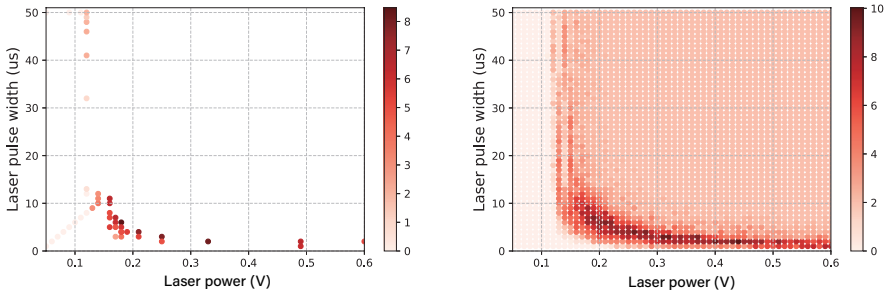
Impact Score for AES The IS-based sensitivity curve is shown in Figure 13a while the full characterization reference is presented in Figure 13b. Similar to the IS distribution shown in Figure 8a, the fault injection is more effective with short laser pulse widths for AES encryption (Figure 13a), as the points with high IS are accumulated at the bottom-right of the graph. Taking Figure 13b as the reference, the IS-based sensitivity curve can cover the overall target behavior effectively with a limited amount of data, thus proving its capability in settings optimization in a short amount of time.



(a) Prediction result.

(b) Error plot when comparing with the full-characterization measured data.

Fig. 12. Prediction result with a deep neural network for AES encryption.



(a) IS: characterization.

(b) IS: exhaustive scan.

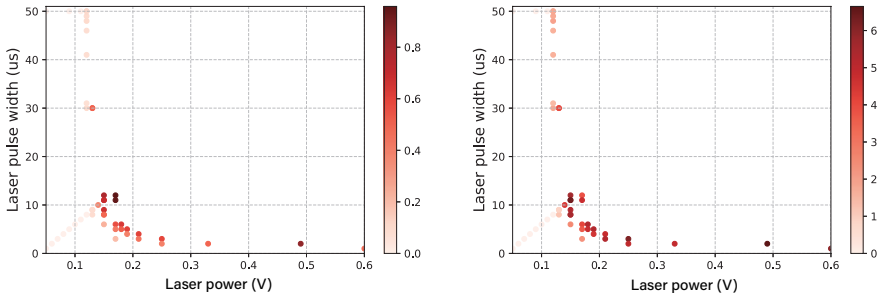
Fig. 13. IS distribution with different fault injection settings for AES encryption.

Transferability of the AES Characterization Results Similar to the experiment performed in Section 5.2, we generated the sensitivity curve with the LOI and IS metrics on a new sample attacking the same locations and using the same laser setup. The results are shown in Figure 14.

From the figures, the LOI and IS distribution are identical to the previous characterization results (Figure 10a and Figure 13a). Therefore, we again show that the characterization result is transferable between different samples of the same TOE. We also conclude from this test on the AES that the fast characterization methodology presented in this paper applies to different attack scenarios.

6 Conclusions and Future Work

In this paper, we present a novel methodology for semi-invasive fault injection attacks that improves the identification (characterization) phase of an attack. This methodology consists of a fast generation of the sensitivity curve and a proper evaluation of the Level of Influence and Impact Score metrics. Instead



(a) LOI: characterization with a different sample.

(b) IS: characterization with a different sample.

Fig. 14. Characterization results with a different sample targeting AES encryption.

of testing FI setting conditions randomly, we start by generating the sensitivity curve, which happens in two phases. First, we find the golden point, which is close to the target LOI and then, we depict the rest of the curve using this point as the reference. Finally, we show how deep learning can be used in fault injection attacks characterization phase where we estimate the full search space by using only a limited number of measurements. In the experimental part, we demonstrated the proposed methodology on running software implementation of DES and AES ciphers. Besides that, we repeat the characterization procedure on a different sample to verify its transferability. Not shown in this paper, the proposed method had been validated for a variety of attack scenarios such as program flow attack and data manipulations. It also showed its effectiveness on other semi-invasive FI techniques such as EMFI and BBI. In the realistic circumstances, attackers can launch our methodology on multiple setups in parallel, which can dramatically boost their attack procedure and performance.

In future work, we plan to further investigate the advantages and limitations of the fast characterization with different fault injection methods, setups, targets, and initial conditions such as temperature and supply voltage. Additionally, we aim to further explore the usage of the neural network in estimating the FI impact on non-measured areas.

References

1. Sergei P. Skorobogatov and Ross J. Anderson. Optical fault induction attacks. In Burton S. Kaliski, çetin K. Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2002*, pages 2–12, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.
2. Rafael Boix Carpi, Stjepan Picek, Lejla Batina, Federico Menarini, Domagoj Jakobovic, and Marin Golub. Glitch it if you can: parameter search strategies for successful fault injection. In *International Conference on Smart Card Research and Advanced Applications*, pages 236–252. Springer, 2013.

3. Stjepan Picek, Lejla Batina, Domagoj Jakobović, and Rafael Boix Carpi. Evolving genetic algorithms for fault injection attacks. In *2014 37th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 1106–1111. IEEE, 2014.
4. Stjepan Picek, Lejla Batina, Pieter Buzing, and Domagoj Jakobovic. Fault injection with a new flavor: Memetic algorithms make a difference. In Stefan Mangard and Axel Y. Poschmann, editors, *Constructive Side-Channel Analysis and Secure Design*, pages 159–173, Cham, 2015. Springer International Publishing.
5. Dan Boneh, Richard A. DeMillo, and Richard J. Lipton. On the importance of checking cryptographic protocols for faults. In Walter Fumy, editor, *Advances in Cryptology — EUROCRYPT '97*, pages 37–51, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg.
6. Oliver Kömmerling and Markus G. Kuhn. Design principles for tamper-resistant smartcard processors. In *Proceedings of the USENIX Workshop on Smartcard Technology on USENIX Workshop on Smartcard Technology*, pages 2–2, Berkeley, CA, USA, 1999. USENIX Association.
7. Sergei Skorobogatov. Optical fault masking attacks. In *2010 Workshop on Fault Diagnosis and Tolerance in Cryptography*, pages 23–29, Aug 2010.
8. Jasper G. J. van Woudenberg, Marc F. Witteman, and Federico Menarini. Practical optical fault injection on secure microcontrollers. In *2011 Workshop on Fault Diagnosis and Tolerance in Cryptography*, pages 91–99, Sep. 2011.
9. Regis Leveugle, Paolo Maistri, Pierre Vanhauwaert, Feng Lu, Giorgio Di Natale, M-L Flottes, Bruno Rouzeyre, Athanasios Papadimitriou, David Hély, Vincent Beroulle, et al. Laser-induced fault effects in security-dedicated circuits. In *2014 22nd International Conference on Very Large Scale Integration (VLSI-SoC)*, pages 1–6. IEEE, 2014.
10. Oscar M. Guillen, Michael Gruber, and Fabrizio De Santis. Low-cost setup for localized semi-invasive optical fault injection attacks. In Sylvain Guilley, editor, *Constructive Side-Channel Analysis and Secure Design*, pages 207–222, Cham, 2017. Springer International Publishing.
11. Eleonora Cagli, Cécile Dumas, and Emmanuel Prouff. Convolutional Neural Networks with Data Augmentation Against Jitter-Based Countermeasures - Profiling Attacks Without Pre-processing. In *Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings*, pages 45–68, 2017.
12. Stjepan Picek, Annelie Heuser, Alan Jovic, Shivam Bhasin, and Francesco Regazzoni. The curse of class imbalance and conflicting metrics with machine learning for side-channel evaluations. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2019(1):209–237, 2019.
13. Jaehun Kim, Stjepan Picek, Annelie Heuser, Shivam Bhasin, and Alan Hanjalic. Make some noise. unleashing the power of convolutional neural networks for profiled side-channel analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2019(3):148–179, May 2019.
14. Antun Maldini, Niels Samwel, Stjepan Picek, and Lejla Batina. Genetic algorithm-based electromagnetic fault injection. In *2018 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*, pages 35–42, Sep. 2018.
15. YongBin Zhou and DengGuo Feng. Side-channel attacks: Ten years after its publication and the impacts on cryptographic module security testing. *IACR Cryptology ePrint Archive*, 2005:388, 2005.

16. Assia Tria and Hamid Choukri. Invasive attacks. In Henk C. A. van Tilborg and Sushil Jajodia, editors, *Encyclopedia of Cryptography and Security*, pages 623–629, Boston, MA, 2011. Springer US.
17. Raghavan Kumar, Philipp Jovanovic, and Ilia Polian. Precise fault-injections using voltage and temperature manipulation for differential cryptanalysis. In *2014 IEEE 20th International On-Line Testing Symposium (IOLTS)*, pages 43–48. IEEE, 2014.
18. Stjepan Picek, Annelie Heuser, Alan Jovic, Simone A Ludwig, Sylvain Guilley, Domagoj Jakobovic, and Nele Mentens. Side-channel analysis and machine learning: A practical perspective. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 4095–4102. IEEE, 2017.
19. Sergei Petrovich Skorobogatov. Semi-invasive attacks: a new approach to hardware security analysis, 2005.
20. Allan H Johnston. Charge generation and collection in pn junctions excited with pulsed infrared lasers. *IEEE Transactions on Nuclear Science*, 40(6):1694–1702, 1993.
21. Dominik Merli, Dieter Schuster, Frederic Stumpf, and Georg Sigl. Semi-invasive em attack on fpga ro pufs and countermeasures. In *Proceedings of the Workshop on Embedded Systems Security*, WESS '11, pages 2:1–2:9, New York, NY, USA, 2011. ACM.
22. Noemie Beringuier-Boher, Marc Lacruche, David El-Baze, Jean-Max Dutertre, Jean-Baptiste Rigaud, and Philippe Maurine. Body biasing injection attacks in practice. In *Proceedings of the Third Workshop on Cryptography and Security in Computing Systems*, pages 49–54. ACM, 2016.
23. Kevin Gurney. *An introduction to neural networks*. CRC press, 2014.
24. Ronan Collobert and Samy Bengio. Links Between Perceptrons, MLPs and SVMs. In *Proceedings of the Twenty-first International Conference on Machine Learning*, ICML '04, pages 23–, New York, NY, USA, 2004. ACM.
25. LeNail. NN-SVG: Publication-Ready Neural Network Architecture Schematics. *Journal of Open Source Software*, 4(33):747, 2019.
26. Eli Biham and Adi Shamir. Differential fault analysis of secret key cryptosystems. In *Annual international cryptology conference*, pages 513–525. Springer, 1997.
27. Christophe Giraud. Dfa on aes. In *International Conference on Advanced Encryption Standard*, pages 27–41. Springer, 2004.