A Fast Characterization Method for Optical Fault Injection

Lichao Wu[1,2], Gerard Ribera[2], and Stjepan Picek[1]

[1] Delft University of Technology, The Netherlands
[2] Brightsight, The Netherlands
L.Wu-4@tudelft.nl, gerardriberas@gmail.com, S.Picek@tudelft.nl

Abstract. Semi-invasive fault injection attacks, such as optical fault injection, 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 on how to find such fault injection parameters. 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 considering targets running DES and AES encryption. Finally, we show how deep learning can help in estimating the full characterization on the basis of a limited number of measurements.

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 a type of attacks that proved to be very powerful in the last decades and where, despite all the efforts, the attacker is still able to 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]. In order to make the attack more efficient and transferable, a characterization of the target of evaluation (TOE) is necessary as the preliminary step of evaluation. Surprisingly, there is no formal approach to this. Manual testing on random parameter combinations is

a common approach to get an impression of the TOE behavior but this approach is not able to provide good coverage of the impact analysis for all the parameter combinations when the investigation is time constrained. Exhaustive search, on the other hand, can be a solution if a full characterization is needed but will consume more time as a trade-off. Finally, techniques coming from the artificial intelligence domain, like genetic algorithms, can work well but face issues like the uncertainty of parameter selection or the high risk of damaging targets when strong FI settings are used.

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 and width, delays (attack time interval), and scan locations. As a complete analysis considering all possible parameters 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 investigate the optimal parameters in every attack scenario, which is not efficient or reliable. For instance, let us assume a small example with 10 000 combinations of the fault injection parameters (100*100) out of which 100 attack locations are selected. If each attack takes one second, 11 days will be spent to finalize an exhaustive search, which is highly time-consuming and could be reduced dramatically by the use of automatic tools. Finally, the existence of countermeasures on hardware and software level can further increase the difficulties in defining other parameters such as delays and scan locations.

In terms of parameter optimization, researchers explored for example genetic [2,3] and memetic algorithms [4]. While such approaches work well for voltage glitching or electromagnetic fault injection, laser fault injection is much more difficult. Indeed, 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 target under attack. Either the change of the setup or the target will result in the change of the optimal parameters.

To standardize the characterization procedure as well as to improve the reproducibility of the fault injection, the development of methodologies that can ensure a proper selection of the tested parameters and the efficiency of an attack is of significant interest. To speed-up the attack parameter identification while considering the setting coverage, we propose a methodology for the fast characterization of the fault injection settings. Our methodology consists of constructing the 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. Finally, we use a deep learning algorithm for the full estimation of the characterization space on the basis of a limited number of measurements.

Our methodology can boost the characterization process while keeping track of useful information, which can eventually lead to 1) a better estimation of the target behavior, 2) a proper selection of the fault injection settings, and 3) a good reference point for future attacks.

1.1 Related Work

Fault injection is a well-researched topic already spanning the range of more than 20 years [5,6]. 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 smart card perturbation attack 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 type of optical fault attacks called fault masking attacks [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]. 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 target. 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 show 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 [11,12,13]. When considering fault injection, there are several works 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 in order 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 in order to characterize the search space for voltage glitching as fast as possible [4]. Maldini et al. used a genetic algorithm for finding fault injection parameters when considering electromagnetic fault injection (EMFI) [14]. 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 optical fault injection 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 optical 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.
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 on the basis of the limited number of actual measurements.

Besides these, we also give detailed experimental results where we investigate our methodology on the AES and DES ciphers.

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 in order to help us better assess the performance of our attack and how to generate/evaluate the sensitivity curve. In Section 5, we discuss our experimental setup and results obtained after attacking a target with the AES and DES ciphers. Finally, in Section 6, we conclude the paper and present some 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 samples. 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 own properties. Similar to the invasive attacks, they require direct access to the chip surface by removing the package, but the passivation layer is kept. Still, a semi-invasive attack can be performed in a reasonably short period of 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 $f : \mathcal{X} \rightarrow \mathcal{Y}$, 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 where the goal is 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 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.

In order 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. Besides the input and output layers, two hidden layers are present in the network. With the help of multiple layers, a deep neural network is able to map complicated low-level details to high-level features progressively. Thus, deep neural networks are able to make a proper estimation of the output, where this adaption process is referred to as deep learning.

In this paper, we use a multilayer perceptron (MLP) algorithm. MLP is a feed-forward neural network that maps sets of inputs onto sets of appropriate outputs. It consists of multiple layers of nodes in a directed graph, where each layer is 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 multilayer perceptron

algorithm 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].

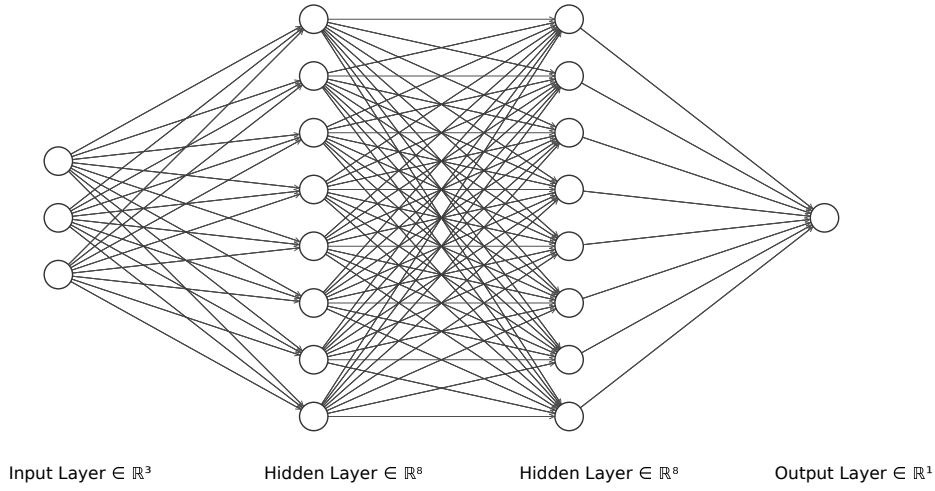


Fig. 1: An example of deep neural network with 2 hidden layers and 8 neurons per hidden layer (created with NN-SVG [25]).

3 Fast Characterization Methodology

A reliable characterization methodology can be used to obtain a quick impression of the influence caused in 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?

The solutions to these problems are summarized with a work-flow shown 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. In order to characterize the target in a fast and correct way, we first generate the sensitivity curve (described in Section 4). Next, we perform an evaluation of the measurements to further investigate the target behavior with different FI settings.

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

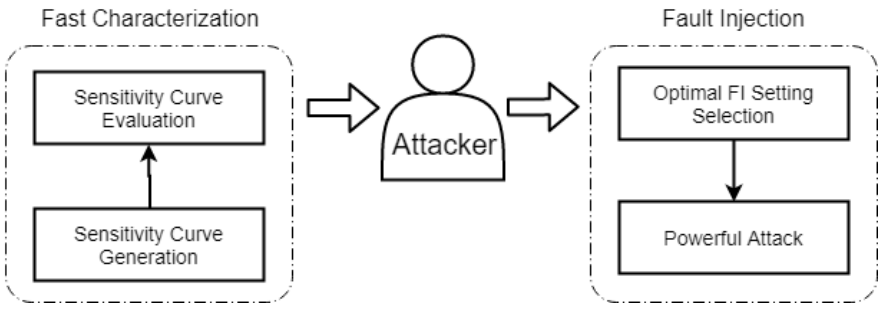


Fig. 2: An attack work-flow with proposed fast characterization methodology.

curve generation. The attack location, for instance, can be delimited 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.

In this section, we start by introducing the notation we use 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, a 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: TOE resets.
3. MUTE: TOE stops communication.
4. CHANGING: TOE returns unexpected values.
5. SUCCESS: TOE returns abnormal but exploitable values.

Note that the proposed methodology can be applied to other fault injection methods that follow the assumption that the strength of the setting is positively correlated to the level of impact on the target. In this paper, we use only optical FI techniques for the experiments. The main attack parameters - the laser input voltage 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. With lower laser settings, for example, the fault injection is less effective and the TOE tends to behave normally, thus having a low influence. In contrast, by increasing the laser settings, there is a higher possibility the TOE is influenced by the injected faults, which will eventually increase its influence in 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.

Impact Score The outputs of the target 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 the weights to each class based on its significance and eventually come up with a score based on each measurement results. As this score directly reflects the effects of the FI with respect to the target behavior, we name this metric Impact Score (IS).

The aim of the Impact Score metric is to show the relevance of the measurements that were acquired during the generation of the sensitivity curve. 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. In contrast, the class NORMAL (indicating the TOE 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

The sensitivity curve consists of a set of fault injection settings that cause a similar impact on the target. While it is possible that the sensitivity curve is not decreasing monotonically as shown in Figure 3, still, regardless of its characteristics, 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.

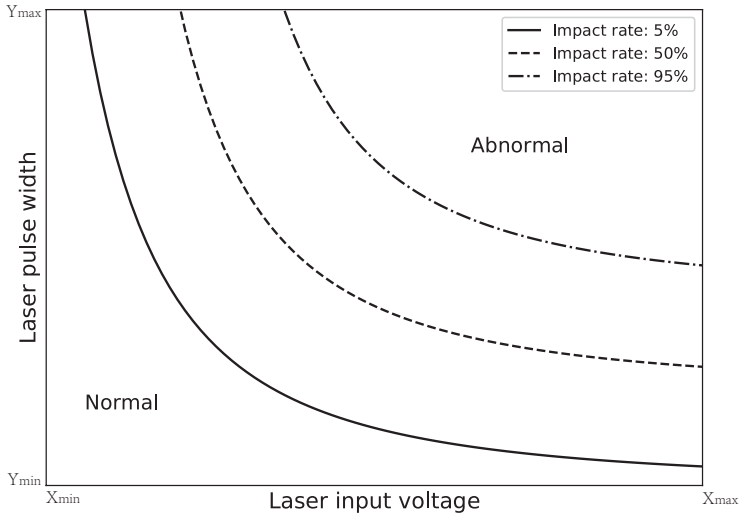


Fig. 3: An example of the sensitivity curves with different LOIs. From here, the normal and abnormal behaviour of the TOE can be estimated.

As shown in Figure 3, with sensitivity curves one can estimate that the injected fault (X and Y axes represents the FI settings) can be ignored at the left side of the curve with 5% LOI; while the target will behave abnormally at the right side of the curve with 90% LOI. Moreover, the figure clearly 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. Indeed, the additional information from the sensitivity curves provides the attacker multiple choices in setting selection: although the LOI is the same, appropriate selection of the FI settings based on the attack scenarios may lead to a more powerful attack. To conclude, the input of the sensitivity curve delimits the number of the parameter combinations to be examined, thus increasing the characterization efficiency. At the same time, with the help of the sensitivity curve, an attacker is able to get a rough overview of the target behavior and make a proper selection of the FI parameters.

4.2 Sensitivity Curve Generation

We use the sensitivity curve for target characterization. Compared with other approaches, the advantages of the sensitivity curve-based characterization are the following:

1. The sensitivity curve defines the natural boundary between the "weak" and "strong" FI settings and thus can be a clear reference for the parameter selection.
2. Since the LOI of a sensitivity curve is defined by an attacker, this resolves the problem of an FI setting selection through a genetic algorithm or random search. While those methodologies perform well for voltage glitching and EMFI, we observed that in optical fault injection, due to a higher risk of damaging the target, one needs to be more conservative in selecting FI settings as, for instance, a too strong pulse will damage the target.
3. The searching of the sensitivity curve can be more time-efficient, as it is achieved with fewer measurements compared to the other methodologies.

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 targets the LOI (C_{target}) defined by an attacker 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. 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.

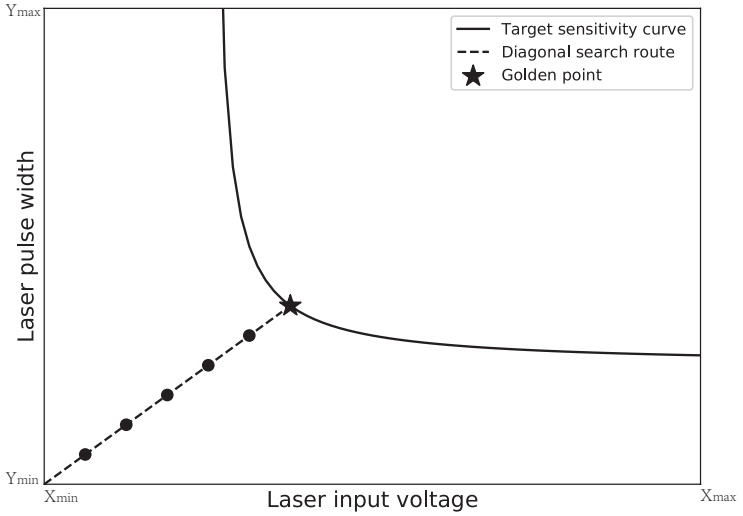


Fig. 4: A demonstration of the diagonal search. The golden point represents the first obtained FI setting with the target LOI.

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 found 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.

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 1) to run in an iterative manner the measurements and 2) calculating the statistics to decide the next couple of settings. Specifically, by varying x while keeping the y obtained by the previous steps, the algorithm is able to keep track of the changing tendency of the target sensitivity curve. Moreover, the usage of the parameters from the

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:         impactRate ← DoTest(x, y)                  ▷ Test with setting (x, y)
8:         if impactRate < C_target + C_tolerance then
9:             y ←BinarySearch(y, Y_max)              ▷ Search with stronger settings
10:        else if impactRate > C_target − C_tolerance then
11:            y ←BinarySearch(y, Y_min)              ▷ Search with weaker settings
12:        data ← data + [x, y, impactRate]
13:    return data                                    ▷ Return all of the tested data
```

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 is getting higher. 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.

Algorithm 2 Step adjustment

```
    function ADJUST_XSTEP(X_step, Y_step, Y_prev, y)
2:      Y_diff ← absolute(Y_prev − y)
        if Y_diff <= Y_step then
4:          return X_step * 2
        else
6:          return X_step / 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. In order to further benefit from the performed measurements, the attacker can use techniques to visualize the data differently with the Impact Score metric in order to obtain an overview of the different setting relevance in FI. Additionally, he can even estimate the non-measured areas 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 response. 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 fault injection attack.

Deep Learning-based Impact Estimation Various advanced techniques can be used to help the attacker to estimate the impact in the non-measured areas. Here, a deep learning-based algorithm is proposed for this task. In many cases, the existence of countermeasures in hardware and software level increases the difficulties in predicting the impact of the FI with different settings. To map the complex relationship between input (FI settings) and output (LOI), machine learning algorithms could be used to perform a rough estimation of the impact that can be caused in the target by using the measurements performed during the sensitivity curve search. Specifically, 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 neural network is able to estimate the LOI with different FI setting, whose accuracy is further evaluated by calculating the offset between the predicted and measured data. In practice, the assessment of attacking the non-measured area is a part of the evaluation and comes from the attacker's decision.

As a remark, the accuracy of the impact estimation highly depends on the quantity as well as the quality of the training data. A sensitivity curve, therefore, should contain sufficient meaningful data in describing the impact of the TOE with different FI settings for an accurate impact estimation. For example, by generating an additional sensitivity curve with different LOI, larger data collection and better setting coverage can be achieved at the same time. In our experiments, we experienced that a deeper and smaller neural network consistently outperformed a shallower but bigger one in terms of impact estimation: by better tuning the network architecture, it is possible to obtain better results with a lower prediction error.

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.

5.1 Experimental Setup

In all our experiments, we use a target based on high-performance 32-bit microcontroller realized in Complementary Metal Oxide Semiconductor (CMOS) technology. No fault injection 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 Data Encryption Standard (DES) cipher and another one targeting the Advanced Encryption Standard (AES) cipher. In both cases, we present a fast characterization that could be used by an attacker to perform the attack in a later stage with the aim to obtain faulty ciphers that can be used to run a DFA attack [26].

The FI setup used to perform the measurements is an optical fault injection setup [27] using an IR light long-pulse laser. The parameters used during the search algorithm are given in Table 1 while the MLP hyper-parameters for the LOI prediction are in Table 2.

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

Table 1: Parameters for the search algorithm.

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

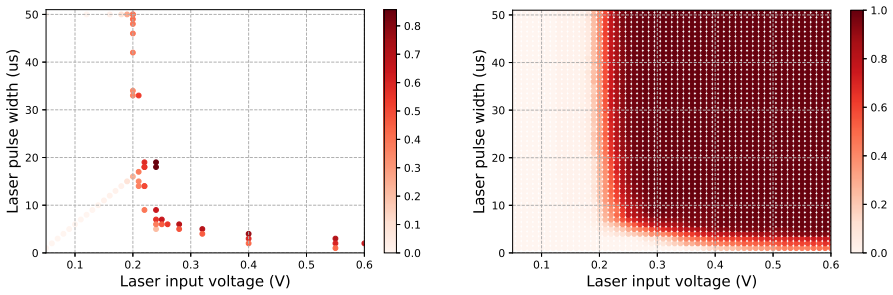
Table 2: MLP hyper-parameters.

5.2 Characterization for 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) and the target locations of the chip are selected by the attacker with a proper understanding of the targeted fault model (e.g., the building blocks with the code stored). These steps remain out of the scope for the proposed methodology. 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, generate the sensitivity curve, followed by the impact estimation using a deep learning algorithm, and the curve regeneration with the Impact Score 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 purpose, 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 5 days of measurement time. As a remark, since the time consumption for each measurement depends on the type of impact that is caused, the duration for the exhaustive measurement is longer than expected.



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

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

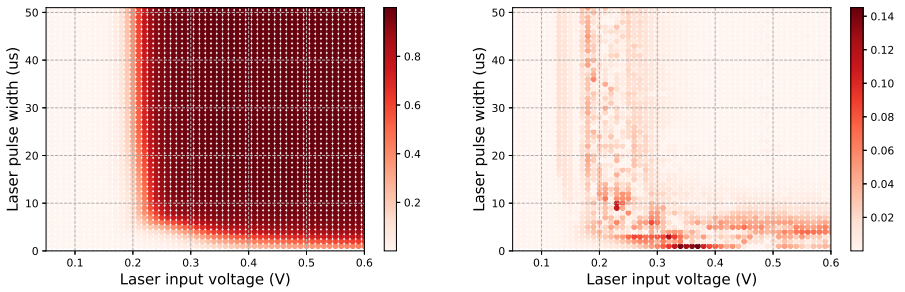
From the result, the outline of the sensitivity curve, which acts as the boundary between “week” and “strong” settings, can be estimated with the measured

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 input voltage is larger than 0.2 V.
2. Similar LOI can be achieved with completely different setting combinations.
3. Laser input 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 appropriated. Nevertheless, an attacker can benefit from these inputs in the next phase of the attack.

The proposed MLP with 6 hidden layers structure (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. As shown in Figure 6a, the prediction result matches the measured data with the majority of the setting combinations. The prediction error plotted in Figure 6b 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 less time than a full-characterization.

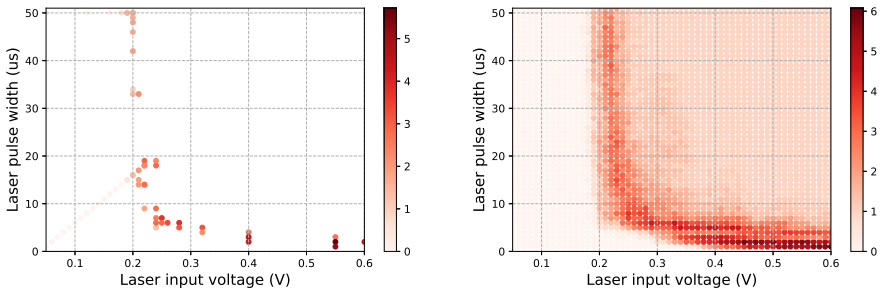


(a) Prediction result using a five-layer neural network.

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

Fig. 6: Prediction result with a deep neural network.

Impact Score for DES In order to further investigate target behavior, Impact Scores are calculated (Figure 7a) based on the measurements performed during the generation of the sensitivity curve. Note that the IS results from the exhaustive scan is shown as the reference (Figure 7b).



(a) IS: characterization. (b) IS: exhaustive scan.

Fig. 7: IS distribution with different fault injection settings.

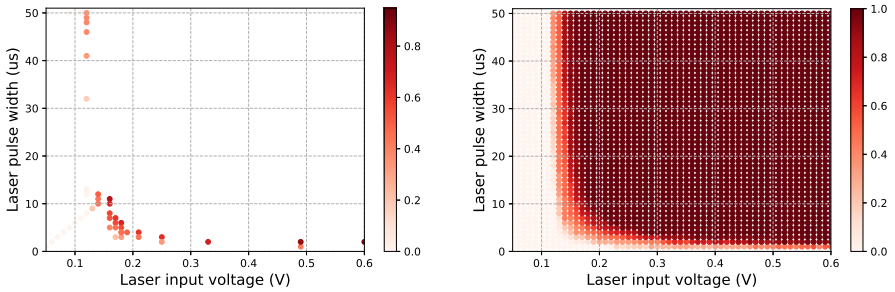
From Figure 7a, a higher IS can be obtained with shorter laser pulse width but stronger input voltage, indicating the high probability in obtaining more significant output in this region. Indeed, this assumption can be proved by Figure 7b with Impact Scores for all setting combinations. Although the IS-based sensitivity curve only covers a few of the setting combinations and the untested optimal settings could still exist, it provides a general layout for the settings with better relevance from the measurements performed, which can eventually lead to a better parameter selection for a later attack stage.

5.3 Characterization for AES Encryption Attack

In order to verify the proposed methodology in different conditions, we perform 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 and the building block to be targeted is selected with a good understanding of the fault model.

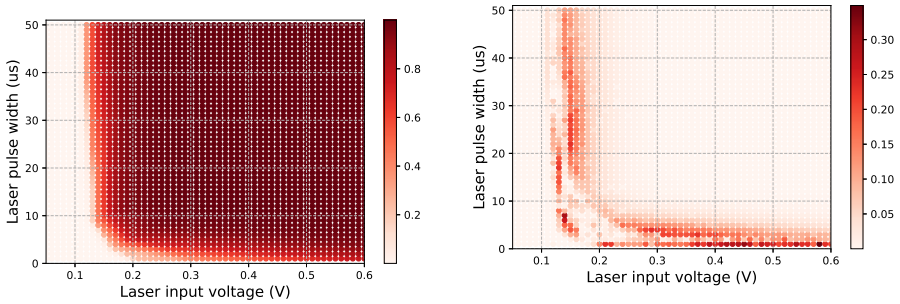
Level of Influence for AES The sensitivity curve is shown in Figure 8a (47 measurements) while its full-characterization counterpart is presented in Figure 8b (3 800 measurements). When comparing this characterization result with the one targeting the DES encryption (Figure 5a), it can be observed that different settings need to be used to obtain comparable LOIs.

The same MLP architecture is used to map the LOI with all the FI setting inputs (Figure 9a) from the data measured during the sensitivity curve generation. When comparing the results with the full-characterization (Figure 8b), we can observe that the LOI tendency is properly estimated. To evaluate the prediction error, the difference between the two is plotted in Figure 9b. Although the 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 potential of the MLP in the LOI estimation is verified.



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

Fig. 8: LOI distribution with different fault injection settings.



(a) Prediction result.

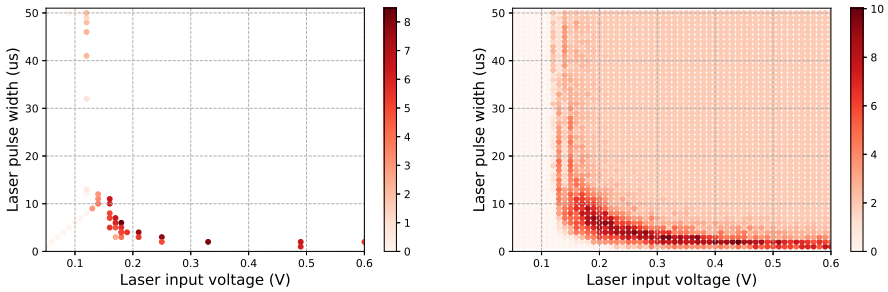
(b) Error plot when comparing with the full-characterization measured data.

Fig. 9: Prediction result with a deep neural network for AES encryption.

Impact Score for AES The IS-based sensitivity curve is shown in Figure 10a while the full characterization reference is presented in Figure 10b. Similar to the IS distribution shown in Figure 7a, the fault injection can be more effective with short laser pulse widths for AES encryption (Figure 10a), as the points with high IS are accumulated at the bottom-right of the graph. Taking Figure 10b as the reference, the IS-based sensitivity curve is able to 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.

6 Conclusions and Future Work

In this paper, we present a novel methodology for optical 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 of testing FI setting conditions randomly, we start by generating the sensitivity curve,



(a) IS: characterization. (b) IS: exhaustive scan.

Fig. 10: IS distribution with different fault injection settings for AES encryption.

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 techniques can be used in fault injection attacks characterization phase where we estimate the full search space by using only a limited number of measurements. Our methodology can be used not only for the fast characterization for a single target/setup but also as a reference to transfer laser settings causing successful attacks to different setups.

In future work, we plan to 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 influence of the neural network hyper-parameter tuning in order to improve the estimation of a full target characterization.

References

1. Skorobogatov, S.P., Anderson, R.J.: Optical fault induction attacks. In Kaliski, B.S., Koç, ç.K., Paar, C., eds.: Cryptographic Hardware and Embedded Systems - CHES 2002, Berlin, Heidelberg, Springer Berlin Heidelberg (2003) 2–12
2. Carpi, R.B., Picek, S., Batina, L., Menarini, F., Jakobovic, D., Golub, M.: Glitch it if you can: parameter search strategies for successful fault injection. In: International Conference on Smart Card Research and Advanced Applications, Springer (2013) 236–252
3. Picek, S., Batina, L., Jakobović, D., Carpi, R.B.: Evolving genetic algorithms for fault injection attacks. In: 2014 37th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), IEEE (2014) 1106–1111
4. Picek, S., Batina, L., Buzing, P., Jakobovic, D.: Fault injection with a new flavor: Memetic algorithms make a difference. In Mangard, S., Poschmann, A.Y., eds.: Constructive Side-Channel Analysis and Secure Design, Cham, Springer International Publishing (2015) 159–173
5. Boneh, D., DeMillo, R.A., Lipton, R.J.: On the importance of checking cryptographic protocols for faults. In Fumy, W., ed.: Advances in Cryptology — EUROCRYPT '97, Berlin, Heidelberg, Springer Berlin Heidelberg (1997) 37–51

6. Kömmerling, O., Kuhn, M.G.: Design principles for tamper-resistant smartcard processors. In: Proceedings of the USENIX Workshop on Smartcard Technology on USENIX Workshop on Smartcard Technology, Berkeley, CA, USA, USENIX Association (1999) 2–2
7. Skorobogatov, S.: Optical fault masking attacks. In: 2010 Workshop on Fault Diagnosis and Tolerance in Cryptography. (Aug 2010) 23–29
8. van Woudenberg, J.G.J., Witteman, M.F., Menarini, F.: Practical optical fault injection on secure microcontrollers. In: 2011 Workshop on Fault Diagnosis and Tolerance in Cryptography. (Sep. 2011) 91–99
9. Leveugle, R., Maistri, P., Vanhauwaert, P., Lu, F., Di Natale, G., Flottes, M.L., Rouzeyre, B., Papadimitriou, A., Hély, D., Beroulle, V., et al.: Laser-induced fault effects in security-dedicated circuits. In: 2014 22nd International Conference on Very Large Scale Integration (VLSI-SoC), IEEE (2014) 1–6
10. Guillen, O.M., Gruber, M., De Santis, F.: Low-cost setup for localized semi-invasive optical fault injection attacks. In Guilley, S., ed.: Constructive Side-Channel Analysis and Secure Design, Cham, Springer International Publishing (2017) 207–222
11. Cagli, E., Dumas, C., Prouff, E.: Convolutional Neural Networks with Data Augmentation Against Jitter-Based Countermeasures - Profiling Attacks Without Preprocessing. In: Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings. (2017) 45–68
12. Picek, S., Heuser, A., Jovic, A., Bhasin, S., Regazzoni, F.: The curse of class imbalance and conflicting metrics with machine learning for side-channel evaluations. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2019**(1) (2019) 209–237
13. Kim, J., Picek, S., Heuser, A., Bhasin, S., Hanjalic, A.: 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) (May 2019) 148–179
14. Maldini, A., Samwel, N., Picek, S., Batina, L.: Genetic algorithm-based electromagnetic fault injection. In: 2018 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC). (Sep. 2018) 35–42
15. Zhou, Y., Feng, D.: Side-channel attacks: Ten years after its publication and the impacts on cryptographic module security testing. IACR Cryptology ePrint Archive **2005** (2005) 388
16. Tria, A., Choukri, H.: Invasive attacks. In van Tilborg, H.C.A., Jajodia, S., eds.: Encyclopedia of Cryptography and Security, Boston, MA, Springer US (2011) 623–629
17. Kumar, R., Jovanovic, P., Polian, I.: Precise fault-injections using voltage and temperature manipulation for differential cryptanalysis. In: 2014 IEEE 20th International On-Line Testing Symposium (IOLTS), IEEE (2014) 43–48
18. Picek, S., Heuser, A., Jovic, A., Ludwig, S.A., Guilley, S., Jakobovic, D., Mentens, N.: Side-channel analysis and machine learning: A practical perspective. In: 2017 International Joint Conference on Neural Networks (IJCNN), IEEE (2017) 4095–4102
19. Skorobogatov, S.P.: Semi-invasive attacks: a new approach to hardware security analysis (2005)
20. Johnston, A.H.: Charge generation and collection in pn junctions excited with pulsed infrared lasers. IEEE Transactions on Nuclear Science **40**(6) (1993) 1694–1702

21. Merli, D., Schuster, D., Stumpf, F., Sigl, G.: Semi-invasive em attack on fpga ro pufs and countermeasures. In: Proceedings of the Workshop on Embedded Systems Security. WESS '11, New York, NY, USA, ACM (2011) 2:1–2:9
22. Beringuier-Boher, N., Lacruche, M., El-Baze, D., Dutertre, J.M., Rigaud, J.B., Maurine, P.: Body biasing injection attacks in practice. In: Proceedings of the Third Workshop on Cryptography and Security in Computing Systems, ACM (2016) 49–54
23. Gurney, K.: An introduction to neural networks. CRC press (2014)
24. Collobert, R., Bengio, S.: Links Between Perceptrons, MLPs and SVMs. In: Proceedings of the Twenty-first International Conference on Machine Learning. ICML '04, New York, NY, USA, ACM (2004) 23–
25. LeNail: NN-SVG: Publication-Ready Neural Network Architecture Schematics. Journal of Open Source Software **4**(33) (2019) 747
26. Giraud, C.: Dfa on aes. In: International Conference on Advanced Encryption Standard, Springer (2004) 27–41
27. company: The name of the company is temporary removed for blind revision