What Do You See? Transforming Fault Injection Target Characterizations

Krček, Marina

DOI
10.1007/978-3-031-22829-2_10

Publication date
2022

Document Version
Final published version

Published in
Security, Privacy, and Applied Cryptography Engineering - 12th International Conference, SPACE 2022, Proceedings

Citation (APA)
Krček, M. (2022). What Do You See? Transforming Fault Injection Target Characterizations. In L. Batina, S. Picek, & M. Mondal (Eds.), *Security, Privacy, and Applied Cryptography Engineering - 12th International Conference, SPACE 2022, Proceedings* (pp. 165-184). (Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics); Vol. 13783 LNCS). Springer. https://doi.org/10.1007/978-3-031-22829-2_10

What Do You See? Transforming Fault Injection Target Characterizations

Marina Krček[✉]

Delft University of Technology, Delft, The Netherlands
m.krcek@tudelft.nl

Abstract. In fault injection attacks, the first step is to evaluate the target behavior for various fault injection parameters. Showing the results of such a characterization (commonly known as target cartography) is informative and allows researchers to assess the target's behavior better. Additionally, it helps understand the performance of new search methods or attacks. Thus, publishing obtained results is essential to provide relevant information for reproducibility and benchmarking, improving state-of-the-art results and general security. Unfortunately, publishing the results also allows malicious parties to reverse engineer the information and potentially mount an attack easier.

This work discusses how various transformations can be used to occlude sensitive information but, at the same time, still be useful for interested researchers. Our results show that even simple 2D transformations, such as rotation, scaling, and shifting, significantly increase the effort required to reverse engineer the transformed data but maintain the interesting data distribution. Consequently, this work provides a method to allow publishers to share more data in a confidential setting.

Keywords: Fault injection · Target characterization · 2D Transformations

1 Introduction

Secure hardware devices should be designed to operate with confidential data so that the information does not leak and cannot be altered by an adversary. While the algorithms running on such devices might be secure, it has been shown that various attacks on hardware can be powerful [6,12]. Such attacks do not attack the algorithms but the weaknesses in the implementation. Those attacks are called implementation attacks and are commonly divided into side-channel and fault injection (FI) attacks. While these attacks are powerful, there are still challenges to improving the attacks to be more efficient.

When considering fault injection, one main challenge is improving the target characterization. Indeed, to mount a successful fault injection campaign, one needs to recognize where the fault should be inserted. Due to the many parameters that need to be tested, this problem can become a very challenging task.

L. Batina et al. (Eds.): SPACE 2022, LNCS 13783, pp. 165–184, 2022.
https://doi.org/10.1007/978-3-031-22829-2_10

New, more powerful attacks are needed to improve state-of-the-art research and contribute to the further security and more efficient evaluation of products. Findings should be shared in a reproducible manner to enable this process. There are cases where the research is done on open public targets, and the results can be shared entirely without restrictions. However, sometimes the data and the actual vulnerabilities of the products must be kept secret as sharing them could pose an economic, privacy, or security threat to target stakeholders. At the same time, it becomes difficult to reproduce the results or even fairly compare them against others without providing sufficient details. Thus, there is a need to enable the community to share the findings publicly without compromising stakeholders.

In this work, we consider sharing data from FI target characterization. We showcase our proposals on data from several types of fault injection - electromagnetic fault injection (EMFI) [19], laser fault injection (LFI) [27], and voltage glitching [3]. Usually, the results of target characterization are shown in a 2D figure with specific FI parameters on the x and y axis. For example, $x - y$ location of the laser or EM probe, or *pulse width* and *intensity* of the laser. We propose several methods to alter the obtained data from the characterization. Accordingly, we allow sharing results publicly while hiding the real vulnerabilities so malicious adversaries cannot directly abuse published information. Publishers can choose the modifications they desire to perform on the data. In this manner, the results of the fault injections and attacks can be published and discussed while the data remains secret. At the same time, transformed data should maintain the original distribution to remain relevant. We propose to use two known metrics to measure the similarity and relation to actual data. Our main contributions are:

1. We showcase that it is easy to recover the exact data points from the cartography (target characterization) figures.
2. We discuss several possible transformations and their effects. We define specific 2D transformations to transform data from 2D plots. We also propose polynomial transformations for transforming more dimensions when not considering the visual representation of the results.
3. We provide two techniques to evaluate the similarity of the original and transformed data and discuss how difficult it would be to reverse engineer the transformed data.

2 Background

2.1 Fault Injection and Target Characterization

Fault injection (FI) can be done physically at the hardware level [5]. Additionally, nowadays, it can also be done on software. However, we focus on fault injection for introducing faults at the hardware level. The idea is to expose the device to various harmful conditions and observe the behavior to determine its response. There are multiple ways to introduce the faults. For example, there are voltage [3] and clock glitching [2,9], temperature variations [26], optical injections [27], and electromagnetic radiation [22,25]. These techniques differ in equipment and cost, precision, and the number of parameters necessary to tune for a successful attack. Once the target is subjected to abnormal conditions (i.e., the external stimuli are introduced), we observe the effects on the device's behavior. Specifically, as analysts, we are interested in at what point the device would fail so that the device can be designed to be more resilient. That is especially important for security-critical devices, such as smartcards. Using previously mentioned techniques for injecting faults, the attacker can change the memory state in a device, cause a mistake in the computation (intermediate values), or skip instructions. Then, the attackers can exploit the faulty results to extract information about confidential data. Examples of these attacks are differential fault analysis (DFA) [4], fault sensitivity analysis (FSA) [15], differential fault intensity analysis (DFIA) [10], and statistical fault attacks (SFA) [8]. Not all faults can be used to reach the malicious goal with these attacks. Thus, the attackers must find a way to inject a fault that can be exploited. Consequently, the fault injection procedure can be divided into two phases: finding faults and using those faults to achieve some (malicious) goal. In this work, we need to be familiar with the first step of finding parameters from the search space that cause faults, i.e., producing the target characterization.

Numerous parameters must be defined for injecting the faults for all the mentioned injection methods. Optimal parameters (parameters that cause the target to show faulty behavior) can be searched manually or with an exhaustive or random search. However, manual testing and a random search are unreliable, as the optimal solutions can be easily overlooked. On the other hand, the exhaustive search is usually very time-consuming. There are many proposed alternatives for finding the optimal set of parameters for different types of fault injection. For example, methods from evolutionary optimization are utilized to improve voltage glitching [7,20,21], EMFI [16], and LFI [13]. Other techniques were also used, e.g., hyperparameter optimization techniques [28] and reinforcement learning [17]. However, while these methods provide a good approximation of specific points (regions), the search space for FI is complex. The issue when using such (intelligent) approaches is that the problem of coverage remains. The obtained optimal parameters are also specific to the setup and target. Finally, the methods need adjustments between different FI techniques. Improvements for conducting target characterization are also proposed in [29]. The methodology is based on finding a sensitivity curve whose generation is fast and compatible with different FI techniques and targets. Additionally, the authors discussed an approach based on deep learning to predict the complete target characterization based on limited data from the sensitivity curve.

2.2 Polynomial Functions

A polynomial with a single indeterminate x can be written in the form:

$$a_n x^n + a_{n-1} x^{n-1} + \cdots + a_2 x^2 + a_1 x + a_0 = \sum_{k=0}^{n} a_k x^k, \tag{1}$$

where $a_0, \ldots, a_n$ are coefficients of the polynomial, and x is indeterminate and can be replaced by any value. For example, x can be substituted with the FI parameters we desire to transform. Thus, we consider a function defined by the polynomial where x is the function's argument and is referred to as a variable:

$$f(x) = \sum_{k=0}^{n} a_k x^k. \tag{2}$$

2.3 Kullback-Leibler Divergence (KLD)

Kullback-Leibler Divergence (KLD) measures how one probability distribution differs from a second, reference probability distribution [14]. For example, one can consider two probability distributions, P and Q. P usually represents the data, the observations, or a measured probability distribution. On the other hand, distribution Q represents a theory, a model, or an approximation of P. KL divergence calculates how one distribution differs from another and is not symmetrical. Calculating the divergence for distributions P and Q would give a different score from Q and P. KLD is the non-negative measure that equals 0 if and only if $P = Q$. For discrete probability distributions P and Q defined on the same probability space, $\mathcal{X}$, KLD is defined as:

$$D_{KL}(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x) \log\left(\frac{P(x)}{Q(x)}\right). \tag{3}$$

2.4 Canonical Correlation Analysis (CCA)

Canonical Correlation Analysis (CCA) is a method of correlating linear relationships between two multidimensional variables [11]. Proposed by Hotelling in 1936, CCA can be seen as the problem of finding basis vectors for two sets of variables. The correlations between the projections of the variables onto these basis vectors are mutually maximized. However, it has been used for measuring the similarity between different neural network layers [18,23]. CCA is invariant to linear transformations and can find shared structures across superficially dissimilar representations. If CCA converges to one, the two compared variables are highly correlated.

3 Motivation and Application

Let us assume that an Evaluator wants to share target characterization data with the general public, including interested researchers in academia, evaluation and certification labs, companies, and malicious parties. The Evaluator can be from academia, an evaluation lab, or a company. They want to either share that they successfully found vulnerabilities in a system previously considered secure or propose new methods for FI target analysis or attack. Sharing all the data helps the community find countermeasures and solutions for the observed vulnerabilities. Consequently, we improve the security of existing systems. Additionally, it is crucial for a fair comparison of the new methods. The data can be used for a public database with realistic data that can help to generalize solutions and benchmark methods and attacks. We can opt for using open public targets, but often these do not represent realistic scenarios. Therefore, the community tends to use targets used by the general public to work in a more realistic and relevant setting. The manufacturer can limit the amount of information shared from such research, which also applies to internal evaluation labs. Sharing data that directly exposes vulnerabilities to malicious parties can raise public concerns and economic threats.

To bridge this gap, we propose to use transformations and explore them in FI analysis. The data in the FI setup is the mentioned target characterization results, parameter values, and device responses to injections with those parameters. This data can be accompanied by target and bench setup information, parameter intervals, and utilized method. Sharing target information and parameter intervals with target characterization data directly reveals vulnerabilities for exploitation and are usually kept secret. With transformations, we motivate to share data at this level as it enables reproducibility and fair comparison. If data is transformed, the attacker cannot directly abuse reported data and speed up the attack process. They will still have to search the parameter space.

In our examples, we consider using brute force for reversing the transformations, as we assume that the authors provide all information on the transformations they applied and fault injection data. However, in a realistic scenario, we expect the author to report that the data is transformed. Still, we do not deem it necessary to report which specific transformations were used as long as data distribution remains close to the original.

4 Proposed Transformations

We consider transformations for altering and hiding results from fault injections. Multiple parameters define the injection during target characterization with any type of FI. For example, in LFI, the parameters can be x, y, *delay*, *pulse width*, and *intensity*. Usually, the results of target characterization are published in a 2D plot with two selected FI parameters on the x and y-axis [13,16,21]. We aim to hide the real vulnerabilities of the target with transformations, but we want to keep the transformed data relevant for publication. We propose 2D transformations on the interesting (vulnerable) points to keep their relative positions (shape they create), but we scale, rotate, and translate the shape.

Since we change only the interesting points, depending on the data, replacing the interesting data with a non-interesting class or randomizing non-interesting points over the whole region will be necessary. We consider both cases in the experiments and explain the choices. Another issue to consider when applying transformations is the possible assumptions that could exist between two parameters that are displayed. Thus, we adjust the transformations so that the resulting transformed data still conforms to the assumptions. For example, analysts expect normal behavior from the device with low absolute values for glitch voltage and length in voltage glitching. Contrary, with high values, we expect the device to reset or stop communication. Interesting responses are usually found between the two regions, and described relative positioning should be kept in transformed data. In our experiments, we use voltage glitching to showcase the changes in the transformations.

These transformations are only used for the selected two parameters shown as the target cartography in a 2D graph. However, we mentioned that all FI types have multiple parameters to set, so if we want to transform all of them and use more than two dimensions, then we propose to use polynomial transformations. These transformations keep the fault class distribution but randomize the data. Therefore, these are unsuitable for cases where we visually must keep the relative position of the classes as in the described voltage glitching case.

In the FI campaign, usually, an interval is defined for each of the parameters with a corresponding step. The step size usually corresponds to the physical properties of the setup. Consequently, we cannot use any value from the interval but only those allowed according to the step size. For the proposed transformations, to ensure we use the specific values, we transform the index of the parameter value instead of the value itself. The code is publicly available[1].

4.1 2D Transformations

Every point in the 2D plot is defined with the x and y coordinates. Note that any two parameters of any FI technique can be set on the x and y-axis. This can be *intensity* and *pulse width* in LFI or EMFI, or x and y location of the laser spot or EM probe on the target. As mentioned, we will rotate, scale, and translate our interesting area (shape) over the target area. We perform rotation with expressions $x_t = x \cos\theta - y \sin\theta$ and $y_t = x \sin\theta + y \cos\theta$. Here, θ is the angle of the rotation. While rotation can be done around any specified point, this formula and what we use in our transformations rotate points around the coordinate system's $(0, 0)$ point. We allow scaling to a minimum of 20% of the entire range for x and y, so the area does not become overly small. For the maximum, we can scale the interesting set of parameters to the entire area. However, we do not necessarily scale equally on both axes, so we can also get the stretching effect. The percentage for the minimum size can be adjusted depending on the real results. To perform the scaling and shifting, we select the starting points (lower bounds) for x and y. The upper bound is then defined with the lower bound and interval size. This way, depending on the lower bound, we have the shift, and depending on the interval, we have scaling.

As mentioned, we need to adjust the transformations for the cases where we must conform to the assumptions we described. Firstly, we limit the angles for the rotation of the interesting area. Secondly, instead of scale and shift, we stretch over both axes and cause a more dense area on other parts. We show this in our experiments with the voltage glitching results, and the reasons are more apparent when we can see the effects visually in the plotted results.

[1] The code is available at https://github.com/marinakrcek/transformations_FI.

4.2 Polynomial Transformations

The explained 2D transformations are used only on two FI parameters shown in a 2D graph within the publication. However, if we want to consider altering the data using all the parameters, then we propose polynomial transformations. We consider these transformations to randomize the non-interesting points or interesting points to lose shape but keep the distribution of fault classes.

We can transform each parameter using a polynomial with different coefficients. We refer to these transformations as *local transformations*. These are used to break the relative positioning of the points. We can also run a *global transformation* that simultaneously transforms all data using the same coefficients for the whole set of parameter combinations. These are used to shift and scale the points. During transformations, the values may get out of bounds, so we have three options for resolving those situations. First, we can *clip* the values, meaning that if the transformed index is out of bounds, we clip it to a lower or higher bound depending on which is closer. Another option is the *modulo* operation (remainder of a division), where if the value is out of bounds, we will calculate a modulo with the number of possible values. Lastly, we have *scale*, where the values are scaled to the original parameter interval. While *modulo* and *clip* can be done immediately after transforming each parameter combination, scaling is done after we transform all the data. This way, we obtain the transformed intervals for the parameters used to scale to the original intervals.

To define the polynomial, the user sets the degree of the polynomial, and the coefficients are selected uniformly at random from user-defined intervals or expressions to define the interval. We can also define a specific polynomial function that controls the output of the transformation, but as we want to randomize the data, we keep the coefficients random. We report the coefficient intervals we used in the presented experimental results. We have a coefficient a_0 not multiplied by the variable x, allowing larger values for this coefficient. We limit the possible values by a maximum of 20% of the allowed values of the parameter. Thus, there is a different interval for the coefficient for each parameter. For global transformation, we use a parameter with the least possible values. For the next coefficient, a_1, we set the allowed interval to $[-2, 2)$. The issue is that the changes will be small with the small indexes, even if the number of allowed values for that parameter is large. We, therefore, allow negative coefficients, as we can still have larger changes depending on the chosen way of handling the out-of-scope values. Other coefficients are defined to achieve lower coefficients for higher polynomial coefficients with the expression $0.5^{(degree-2-i)}$, where *degree* is the defined polynomial degree. We add the term -2 as intervals for coefficients a_1 and a_0 are already defined. i is the counter from $degree - 2$ to zero. These coefficients must get smaller as x has larger exponents because, in our case, x is an index, a positive value that can get rather large as the exponents get larger.

We noticed that the polynomial of degree 1 is sufficient, and larger polynomial degrees do not change the data in any other different pattern than visible with the polynomial of degree 1. The difference is that the changes are more significant, which is quite prominent with clipping, as more points get clipped to maximum or minimum values for the parameters. We tested several other combinations of the coefficient intervals and expressions with smaller and larger values. Our search is not exhaustive, but we noticed similar behavior with larger coefficients as with larger polynomial degrees. Also, the benefit of using the global transformation after the local one is that the data is not spread over the whole parameter 2D space but usually occupies a smaller region of a rectangular or oval shape.

5 Utilized Data Examples

To allow evaluation of the proposed transformations, we need relevant examples of target characterizations. We do not use real confidential data because we cannot show the original data and its transformed data. Instead, we use published work and one simulated example.

First, we use an example from [16] with a graph of Electromagnetic FI (EMFI) showing x-y locations on the target and corresponding fault classes. The authors use RESET, NORMAL, CHANGING, and SUCCESS fault classes. Since we investigate different examples, we use the fault class names MUTE, PASS, CHANGING, and FAIL, which correspond to the mentioned fault classes. As we did not have the original data, we extracted it from the pixels of the image. Similarly, attackers could obtain results from published figures to get precise data points. Note that the attacker can take the interval and search only in that area, which is more efficient than mounting a complete characterization. However, extracting from the pixels is more specific and speeds up an attack. With transformations, we want to prevent this. Additionally, we show another example that corresponds well with possible LFI or EMFI campaign results, showing x-y locations on the target in the 2D plots. For this example, we also show 3D plots with intensity on the z-axis. The third example is somewhat different, where glitch voltage and length are on the x and y-axis. The data is obtained in the same manner as for the EMFI data example from [21]. This example represents parameters for which analysts have some assumptions. Specifically, in this case, the assumption is that we expect normal behavior from the device (PASS) with a low values combination of those parameters. Contrary, with high values for that combination of parameters, one would expect the device to reset or stop communication (MUTE). The interesting FAIL responses are usually situated on a border between the two regions which analysts try to find during characterization. Another example of such parameters would be laser intensity and pulse width for LFI.

Reverse Engineering Data Points from Figures. There are online tools, such as Webplotdigitizer[2] [24] or PlotDigitizer[3] [1], where one can upload an image, and after aligning the x and y axes, it is possible to extract the information

[2] https://automeris.io/WebPlotDigitizer/.

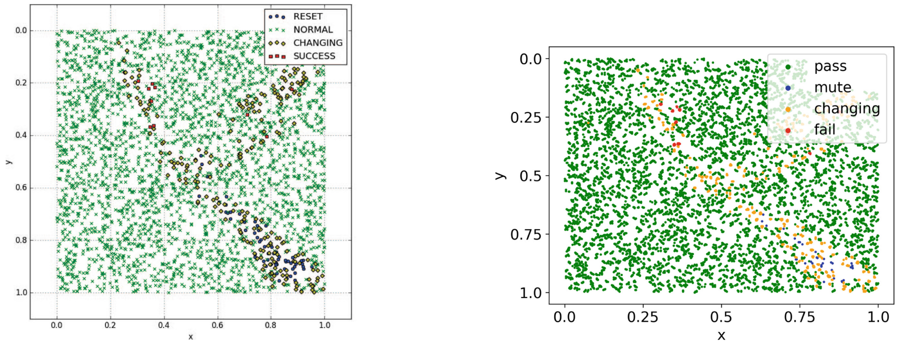
[3] https://plotdigitizer.com/.

about certain points from the plot. However, we used Python Imaging Library to read the pixels as it was easier to save the data for later transformations. Each pixel defined with its location has an RGB (Red Green Blue) code - an array with three values for determining the color. From the legend, we can learn the color of each fault class in the plot. From the range information on each axis, we can scale the data from pixels to the actual scope of the parameters. In this manner, we obtain the parameter values from the image and the device's response per parameter combination.

6 Experimental Results

6.1 Electromagnetic Fault Injection (EMFI) Case

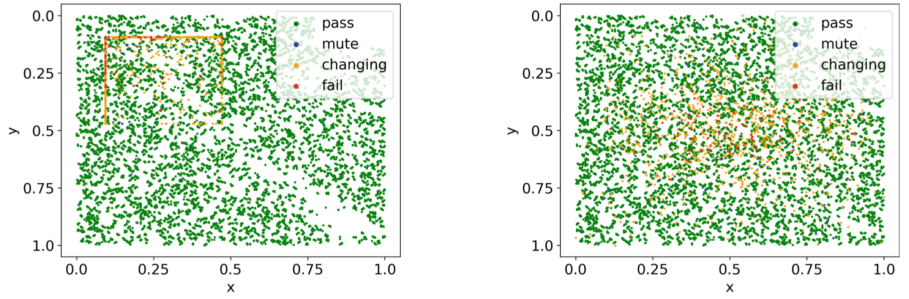
Transformations. We start with the EMFI case, where the authors presented information about the device, its size, used intervals, and the obtained results [16]. The results from the original paper are presented in Fig. 1a. As previously explained, we extracted the data from pixels in the image, and the result is visible in Fig. 1b. As mentioned, we recommend polynomial transformations for randomizing the data, and they are specifically useful for more than two dimensions when we do not care about visual results. Nevertheless, we first show results using polynomial transformations to showcase their issue when using them for the visual representation of the results. Transformed data is visible in Fig. 2. We transform the interesting points while the non-interesting (PASS) remain the same. In Fig. 2a, we show a polynomial of degree 1 with the clip method for values out of bounds. Here, the original interesting area is visible as an empty area as we did not replace the points, neither we alter the rest of the non-interesting points. The clip method is noticeable in the edges of the rectangular shape. In the setting without global transformation, the values are on the borders of the plot. The global transformation translated and scaled the interesting area after local transformations. With larger polynomial degrees (2 and 3), more values



(a) Original plot from [16].

(b) Extracted data from Figure 1a.

Fig. 1. Original cartography from [16] and extracted data from the image.



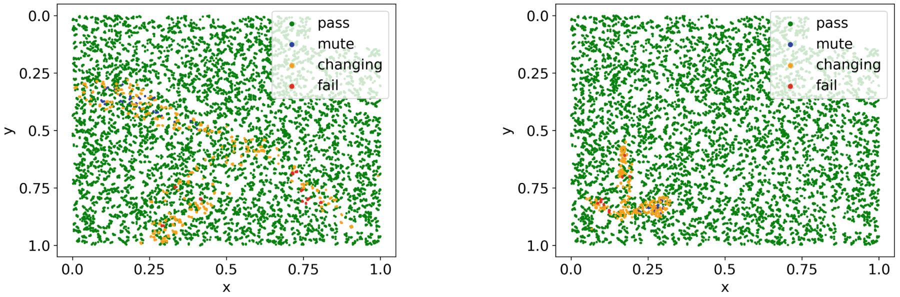
(a) Polynomial of degree 1 with clip method and global transformation.

(b) Polynomial of degree 1 with scale method and global transformation.

Fig. 2. Polynomial transformations on cartography shown in Fig. 1.

are clipped and end in the image's corners. Next, we show the results with the scale method in Fig. 2b, which rounds the interesting points around the central point in the image. Also, we replace the interesting points in the original data with a non-interesting class as we want to cover the empty space in the plot that indicates where the interesting points were located. The difference between results with and without global transformations is the translation of the central point and scaling. The points converge more to the central point with a higher polynomial degree and larger coefficients. We also tested the modulo method with the same example. The transformation results with modulo are that the data is fully randomized over the whole area if global transformation is not used. Similarly, global transformation can shift and scale the area, and the interesting area can become a smaller rectangular shape. The resulting shapes of the interesting area are very different from the original data. Still, if we do not consider the visual shapes, we can use the transformations on more dimensions for statistical analysis.

We now show two different 2D transformations on the extracted data from [16]. First, we have the transformation results shown in Fig. 3a we refer to as transformation T1. The issue with the result of T1 is the overlap with the interesting area in the original cartography. An example without such an overlap is preferred and visible in Fig. 3b as transformation T2. Non-interesting points replace the original interesting area as before. Transformation preserves the shape from the original cartography, but it is rotated, scaled, and moved to another region. Thus, the attacker could focus on the area shown in the figure and miss the actual interesting area. Finally, the actual values of x and y on corresponding axes are hidden by normalizing the data. Without knowing the parameter intervals, we do not know if the whole target was tested or only a specific smaller part.



(a) T1: Angle of rotation is 165°, x is in the interval [0.01, 0.99], and y in [0.28, 0.99].

(b) T2: Angle of rotation is 310°, x is in the interval [0.05, 0.32], and y in [0.55, 0.93].

Fig. 3. 2D transformations on cartography shown in Fig. 1.

Note that x-y target characterizations can end in different unique shapes. Therefore, one can consider that knowing the shape can still help the attackers make more efficient attacks. So, depending on the wanted level of security, we can change the shape with local polynomial transformations or with more specific transformations for different shapes.

Reversing Transformed Data. Now, we discuss how an attacker could find the correct transformation presented in a certain work. We consider that the attacker knows what transformations are used, and we also assume that the attacker knows the intervals for the parameters. We investigate how many possible transformations there are and how long it would take to reach the original cartography with a brute-force approach.

Since we use rotations on the interesting area, we have 360 possible rotations. For scale and shift, the number of possibilities depends on the number of possible values of the parameters for the x and y axes. The number of possible transformations is calculated with the following formula for each parameter:

$$\frac{n(n+1)}{2}, n = \lceil 0.8 \cdot nb\_values \rceil + 1. \tag{4}$$

nb_values is the number of possible values for a specific parameter. As previously explained, we select the interval size and the lower bound to define the shift and scaling. The possibilities for the interval size are between 20% of the possible values and all possible values. Depending on the selected size, there are more or fewer possibilities to set the lower bound of the new interval. For example, if we uniformly at random select that the size of the interval is 20% of all possible values for that parameter, then the number of possibilities for the lower bound is the highest - 80% of the total number of possible values for the parameter. If, on the other hand, the selected size is all the possible values of the parameter, then there is only one possibility for choosing the lower bound. In the end, we have a sum of options calculated with the expression above.

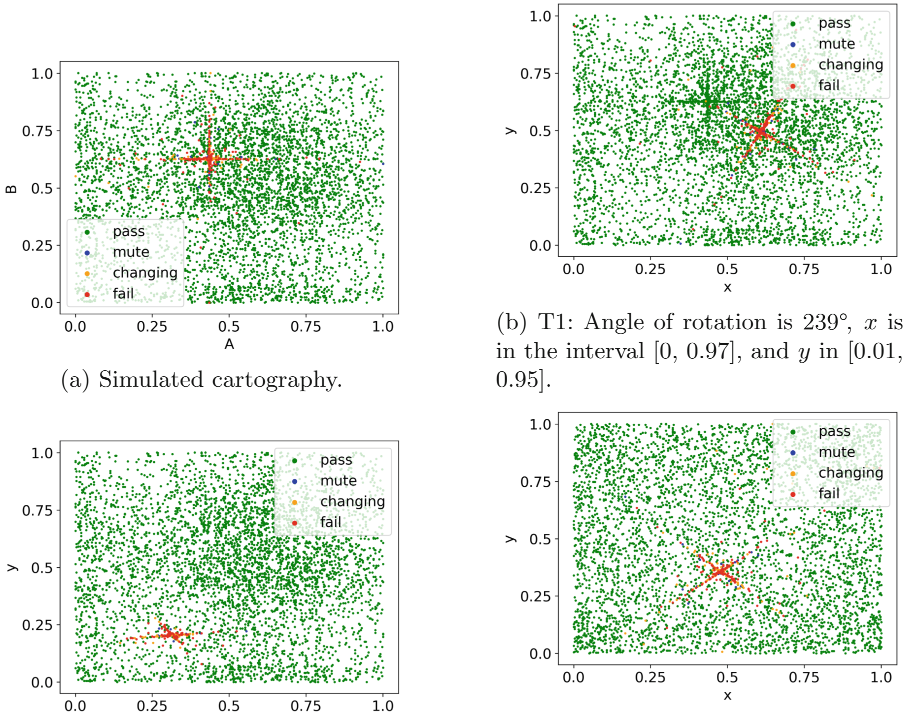
In the case of EMFI cartography, with 481 possible values for both x and y, we have 74 691 possibilities for each, which in combination gives $\approx 5.58 \times 10^9$ options. With rotations, we have $\approx 2.01 \times 10^{12}$ possible transformations in this setting. If it takes 1 ms to test one possible transformation, it will take around 63 years to test all combinations. Therefore, if we consider the attack setting as described, it would take too long for the attacker to test all transformations and find the correct one in a reasonable time.

6.2 Simulated Case

We test the transformations on another example of a specific shape found with a fault injection campaign. We consider it to represent the x-y cartography of the EMFI or LFI campaign. The example does not correspond to any target or real cartography but is a good example as it highlights possible issues with the current transformations.

Transformation. We refer to the cartography presented in Fig. 4a as the original cartography, and we transform the data shown in that plot. We initially transform the data in the same way as in the previous example, and the result is visible in Fig. 4b. We replace the originally interesting area with a non-interesting area. However, since the area has a specific shape and many interesting points when replaced by a non-interesting fault class, we still see where the previous location was. In the following transformation in Fig. 4c, we do not replace the interesting area with a non-interesting fault class. Additionally, the interesting area is far from the original, interesting area, which is the desired result. However, we still notice that the non-interesting points are denser in the area close to the originally interesting area, which could help attackers find the real vulnerabilities. Since polynomial transformations are good for randomizing the data, we perform the local polynomial transformation of degree 1 with the modulo method, but only for the non-interesting fault class. We selected modulo as it was shown in our previous experiments that it had the best ability to spread the points over the entire target area. The result is a plot in Fig. 4d, where we see that the points are randomized over the whole target area, and there are no particularly dense areas to attract attention. In this transformation, the new interesting area is again not close to the actual interesting area hiding the real vulnerable locations. Here, we do not disclose the coefficients of the polynomials as they are selected uniformly at random from previously described intervals for each non-interesting x-y combination.

Reversing Transformed Data. In this setting, we have 32 896 interval combinations for x, and 61 075 for y, which equals $\approx 2.01 \times 10^9$ combinations in total. Again, we add the rotations and reach $\approx 7.23 \times 10^{11}$ combinations. In this case, we would need 22.94 years to test all transformation combinations if testing one transformation takes 1 ms. While we need less time to test all the transformations, it is still unreasonable to consider brute force.



(a) Simulated cartography.

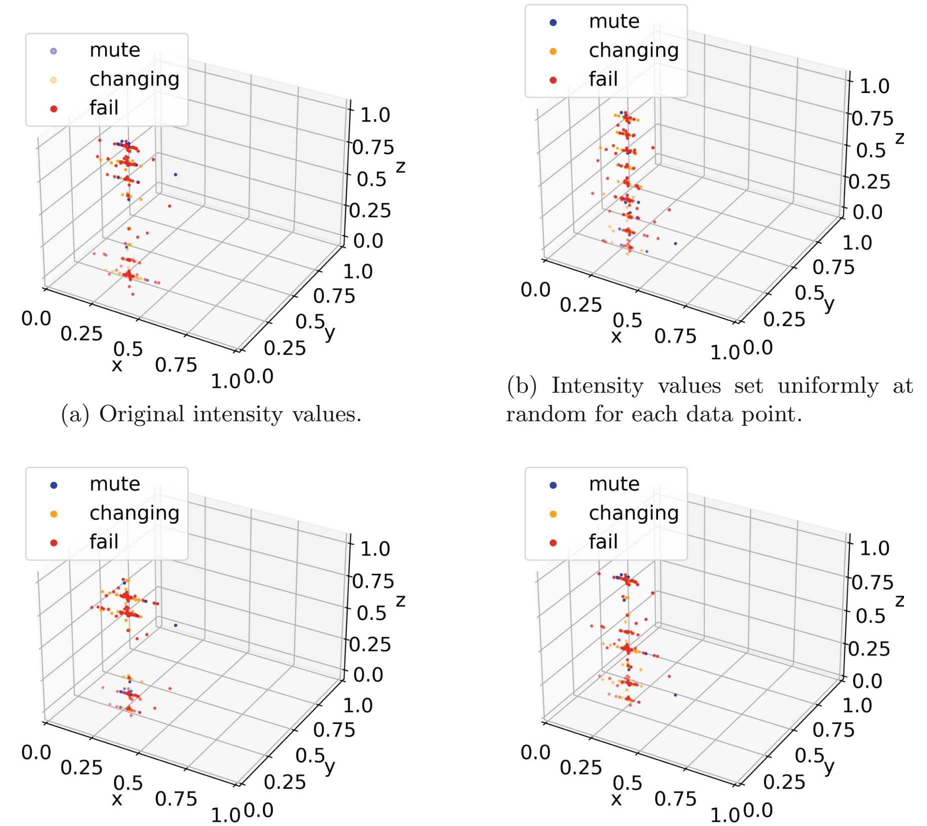
(b) T1: Angle of rotation is 239°, x is in the interval [0, 0.97], and y in [0.01, 0.95].

(c) T2: Angle of rotation is 15°, x is in the interval [0.14, 0.56], and y in [0.08, 0.28].

(d) T3: Angle of rotation is 134°, x is in the interval [0.17, 0.99], and y in [0.01, 1]. The polynomial transformation is used for non-interesting points.

Fig. 4. Simulated cartography with its transformations.

3D Plot. Additionally, we show that we can transform data for figures that display three different parameters in a 3D plot. We use transformed data shown in a 2D plot in Fig. 4c and add the intensity to the z-axis. Figure 5a shows the data in 3D with the original intensity values without showing the non-interesting points for better visibility of the interesting area. As with other parameters, we also normalize the data for the z-axis. Hiding the actual intensity values by normalizing them could be enough. If we do not specify the range we used and disclose the information about the bench and the laser, it would be hard for an attacker to reverse the intensity values. However, we can randomize the intensity as well. Figure 5b shows the transformation of intensity in a way that for every point, a new random intensity was selected. On the other hand, in Fig. 5c, we map all possible values of the intensity to another intensity value. Then, the original intensity value gets replaced by the preselected random intensity value for every point. The values of the intensity can repeat in this setting. Thus, we add another option where we create unique mappings and use those to alter the intensity values. This transformation is visible in Fig. 5d.



(a) Original intensity values.

(b) Intensity values set uniformly at random for each data point.

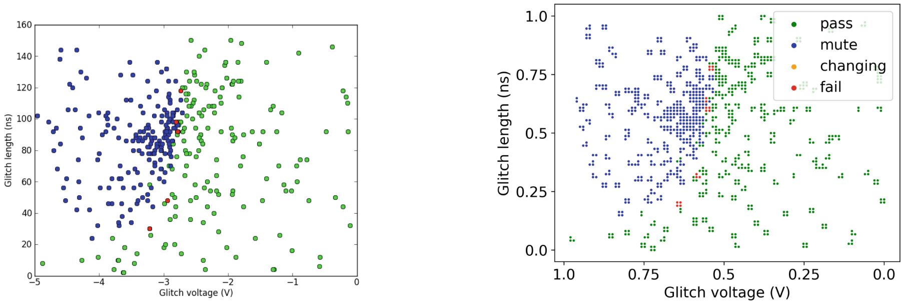
(c) Intensity values mapped to intensity values with possible repetition.

(d) Intensity values mapped to unique intensity values.

Fig. 5. T2 (Fig. 4c) of simulated cartography with transformations for the intensity on z-axis.

6.3 Voltage Glitching Case

The last use case is based on voltage glitching experiments presented in [21]. The original results are in Fig. 6a. There is glitch voltage on the x-axis, and on the y-axis is the glitch length. In this case, contrary to x-y locations, there are generally applicable assumptions for the target's responses depending on the glitch voltage/length values we already described. The analysts search for the boundary between the two regions. Therefore, we want to adhere to the assumptions by keeping the relations with transformed data but hiding the actual border between the classes where the device behaves as expected (PASS) and resets or stops communication (MUTE).



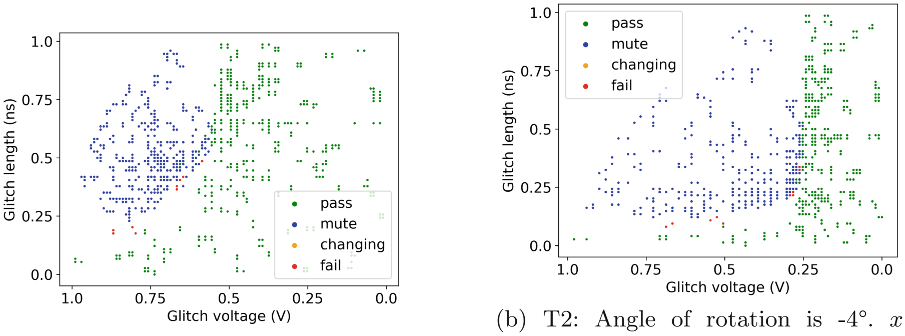
(a) Original plot from [21].

(b) Extracted data from Figure 6a.

Fig. 6. Original cartography from [21] and extracted data from the image.

Transformation. First, we extract the data from the original plot (Fig. 6a), and the result is visible in Fig. 6b. Note that the real parameter values are visible in the original plot, but we display plots with normalized values. Considering the assumptions, we care about absolute values, so the value −5 for the glitch voltage is replaced with 1.

The issue with the transformation we used for x-y characterization is that the boundary between interesting and non-interesting areas would not align with the mentioned assumptions. Suppose we can rotate the interesting area with any of the 360 angles. In that case, we can get a transformation where the border is not between the regions but at the plot's far left. On the other hand, if we rotate all the points, we can get a plot indicating the opposite response of the target - lowest values lead to MUTE and highest to PASS class. Moreover, it might be enough for these types of parameters to normalize the data. If we do not specify the range we used, it would be hard for an attacker to reproduce the injections. However, we still slightly adjusted the transformations by limiting the possibilities of previous transformations to conform with the assumptions for the glitch voltage-length parameters. Firstly, we do not allow all possible angle rotations but only from $[-80°, 30°]$, which we defined using the trial-and-error approach by visually checking if the assumptions still hold. The parameter combinations with FAIL fault class stay close to the border with a non-interesting area and do not invert to the opposite side using defined rotations. Previously, if we scaled and shifted only the interesting area, we lost the relative positioning of the MUTE and PASS classes. Instead, we make data points more dense or sparse by splitting the data below and above certain values on the x and y-axis. Then, we select new splitting x and y values and scale the data. Scaling is done so that the points above the first selected value remain above the newly selected splitting value and analogously for points below the selected values. Let us assume we selected a value x_1 and then x_2. In this case, the points below x_1 will be scaled from 0 to x_2. The value of x_2 can be lower or higher than the x_1. If x_2 is lower than x_1, the points will be denser; otherwise, the points will be more stretched as the interval increases. The results of these transformations are visible in Fig. 7. We can see that the relative positions of the different areas remain in both figures. In the T1 transformation in Fig. 7a, the FAIL points are close to the original border but rotated so that lower glitch length leads to those points. In the T2 transformation, the MUTE area is stretched, while the PASS area is denser since the border is moved to the right. As the data is stretched, the points become sparse in some areas. The exploration with algorithms is usually random over the whole search space, so one cannot expect such sparse testing in specific regions. Thus, we need to consider this for publishing the results. However, an algorithm used in this example converges to FAIL outcomes. The sparseness is explained by convergence in the algorithm, also visible in the original cartography.



(a) T1: Angle of rotation is -53°. No change on x. y moved from 0.36 to 0.44.

(b) T2: Angle of rotation is -4°. x moved from 0.67 to 0.31 and y from 0.44 to 0.27.

Fig. 7. Transformations of the glitching example (Fig. 6a).

Reversing Transformed Data. With the described transformations, we have even fewer possible transformations. The reason is fewer possible values for the parameters and limited possibilities because the transformations need to conform to the assumptions. We allow 110 possible rotations, and the splitting points are between 20% and 80% of all possible voltage or glitch length values. From 100 possible values for voltage, we allow 60, and from 75 possibilities for length, we have 45. In total, that is 7.29×10^6 possibilities, and after adding the rotations we have 8.019×10^8 possible transformations. It will take 9.28 days to try all combinations if one takes 1ms. That is much less time to test all combinations than in the previous examples. However, as already mentioned, hiding the parameter ranges, in this case, could be enough.

6.4 Evaluating the Effect of Transformations

To evaluate the effect of transformation, we use Kullback-Leibler divergence (KLD) and Canonical Correlation Analysis (CCA). KLD and CCA evaluate how similar the transformed data is to the original data. KLD is used to compare the distribution of fault classes and data points between original and transformed data, while CCA indicates the level of correlation between the two data sets. Since our transformations use randomness, with CCA, we measure if the transformed data has been randomized to a point where there is almost no correlation with the original data. While this was more critical for polynomial transformations, we kept it for the 2D transformations. Note that the implementation of CCA is taken from the public GitHub repository[4] [18,23].

Using the notation from the previous KLD definition, we consider the data from true cartography (target characterization) as the reference probability P and the probability of the transformed data as Q. We compute how the distribution from the transformed data differs from the true data for fault classes and utilized parameter values. The probability distribution for KLD is obtained by finding the frequency of each possible value for parameters in true and transformed data. We calculate KLD for each parameter and show the mean KLD in Table 1. Similarly, we calculate the KLD for fault class distribution. On the other hand, since we replace the original interesting area with a non-interesting fault class in some examples, we effectively add the newly transformed data to the existing one. For this reason, there is a different number of data points in the original and transformed data, and to calculate the CCA, we need to have the same number of data points. In most cases, we only transform the interesting area, so we calculate the CCA only on the interesting points. However, when possible, we show CCA on all data points, which is visible in the same Table 1. Occasionally, there could be overlaps and, with that, a possible change in fault class distribution, but this remains low, as visible by the KLD in all cases. The difference between the data's original and transformed distribution of parameter values remains low in the EMFI example. In the simulated example, with each transformation, the KLD increased. In T1, the original interesting area was replaced with non-interesting points. Then in T2, we removed this, and KLD increased. Lastly, we used polynomials to randomize the PASS fault class, resulting in a higher KLD because all the points have been modified. However, the worst situation is in the example with voltage glitching, as KLD is very high. This example's number of data points is lower than in other examples. Thus, many possible points are not tested, so the difference in the value distributions is high. CCA converges nicely to one, meaning original and transformed data are highly correlated. However, with the simulated example, when we calculate CCA for the interesting points, it is below 0.5. For T2, CCA calculated on all data points is close to 1 as points of the PASS class remain the same. On the other hand, with T3 transformation, CCA is almost zero because as the PASS is randomized, all data points are different. The issue might be the specific shape and the number of altered points. Visual inspection still provides the best indication for publishing, but these metrics offer good insight into the performed modifications. Metrics show that the transformations keep the fault class distributions and remain correlated with the original data.

[4] https://github.com/google/svcca.

Table 1. Kullback-Leibler Divergence (KLD) and Canonical Correlation Analysis (CCA). By default, CCA is calculated only on data points of interesting fault classes.

	EMFI			Simulated				Voltage glitching			
	KLD for classes	KLD for parameter values	CCA on interesting data	KLD for classes	KLD for parameter values	CCA on interesting data	CCA on all data points	KLD for classes	KLD for parameter values	CCA on interesting data	CCA on all data points
T1	0.0039	0.0138	0.9999	0.0069	0.0053	0.4243	/	0.0035	10.679	0.9953	0.9221
T2	0.0186	0.0275	0.9999	0.0045	0.2172	0.2895	0.9355	0.0035	4.6081	0.9851	0.8684
T3	/	/	/	0.0001	0.3503	0.3738	0.0275	/	/	/	/

7 Conclusion and Future Work

This work provides several techniques for transforming the target characterization results to hide sensitive information. Indeed, we show that from a figure (a typical representation of a characterization experiment), one could easily obtain the exact data points leading to a fault. We discuss various transformations and analyze the results for three different scenarios showing that using transformations significantly hinders the possibility of reverse-engineering the data from graphs. Additionally, we show that our transformations maintain the correct information about the data distribution and are highly correlated with the original data, making the transformed figures relevant. We show these transformations provide additional layers of hiding confidential data. We discuss potential cases where such transformations could be useful, and with that, we try to motivate Evaluators to share more data as it can lead to improved benchmarking and, consequently, the security of different systems against fault injections.

Proposed transformations are rather simple, which makes them easy to apply. However, more research should be done to provide guarantees on the effort to reverse the data. Furthermore, we aim to explore how to make automated transformations. Current experiments still require an expert with knowledge about the nature of parameters to select appropriate transformations. Building a rule-based system that can transform the data while maintaining the relevant assumptions would be interesting.

Acknowledgements. We thank the reviewers for their time and feedback, especially shepherd Shivam Bhasin.

References

1. PlotDigitizer: Version 2.2 (2022). https://plotdigitizer.com
2. Agoyan, M., Dutertre, J.-M., Naccache, D., Robisson, B., Tria, A.: When clocks fail: on critical paths and clock faults. In: Gollmann, D., Lanet, J.-L., Iguchi-Cartigny, J. (eds.) CARDIS 2010. LNCS, vol. 6035, pp. 182–193. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-12510-2_13
3. Aumüller, C., Bier, P., Fischer, W., Hofreiter, P., Seifert, J.-P.: Fault attacks on RSA with CRT: concrete results and practical countermeasures. In: Kaliski, B.S., Koç, K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 260–275. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-36400-5_20

4. Biham, E., Shamir, A.: Differential fault analysis of secret key cryptosystems. In: Kaliski, B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 513–525. Springer, Heidelberg (1997). https://doi.org/10.1007/BFb0052259
5. Boneh, D., DeMillo, R.A., Lipton, R.J.: On the importance of checking cryptographic protocols for faults. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 37–51. Springer, Heidelberg (1997). https://doi.org/10.1007/3-540-69053-0_4
6. Breier, J., Hou, X.: How practical are fault injection attacks, really? Cryptology ePrint Archive, Paper 2022/301 (2022). https://eprint.iacr.org/2022/301
7. 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: Francillon, A., Rohatgi, P. (eds.) CARDIS 2013. LNCS, vol. 8419, pp. 236–252. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-08302-5_16
8. Fuhr, T., Jaulmes, E., Lomné, V., Thillard, A.: Fault attacks on AES with faulty ciphertexts only. In: 2013 Workshop on Fault Diagnosis and Tolerance in Cryptography, pp. 108–118. IEEE (2013)
9. Fukunaga, T., Takahashi, J.: Practical fault attack on a cryptographic LSI with iso/iec 18033–3 block ciphers. In: 2009 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC), pp. 84–92. IEEE (2009)
10. Ghalaty, N.F., Yuce, B., Taha, M., Schaumont, P.: Differential fault intensity analysis. In: 2014 Workshop on Fault Diagnosis and Tolerance in Cryptography, pp. 49–58. IEEE (2014)
11. Hardoon, D.R., Szedmak, S., Shawe-Taylor, J.: Canonical correlation analysis: an overview with application to learning methods. Neural Comput. **16**(12), 2639–2664 (2004). https://doi.org/10.1162/0899766042321814
12. Kocher, P., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M. (eds.) Advances in Cryptology – CRYPTO 1999. CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48405-1_25
13. Krček, M., Fronte, D., Picek, S.: On the importance of initial solutions selection in fault injection. In: 2021 Workshop on Fault Detection and Tolerance in Cryptography (FDTC), pp. 1–12 (2021). https://doi.org/10.1109/FDTC53659.2021.00011
14. Kullback, S., Leibler, R.A.: On information and sufficiency. Ann. Math. Stat. **22**(1), 79–86 (1951)
15. Li, Y., Sakiyama, K., Gomisawa, S., Fukunaga, T., Takahashi, J., Ohta, K.: Fault sensitivity analysis. In: Mangard, S., Standaert, F.-X. (eds.) CHES 2010. LNCS, vol. 6225, pp. 320–334. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15031-9_22
16. 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), pp. 35–42. IEEE (2018)
17. Moradi, M., Oakes, B.J., Saraoglu, M., Morozov, A., Janschek, K., Denil, J.: Exploring fault parameter space using reinforcement learning-based fault injection. In: 2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W), pp. 102–109. IEEE (2020)
18. Morcos, A., Raghu, M., Bengio, S.: Insights on representational similarity in neural networks with canonical correlation. In: Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R. (eds.) Advances in Neural Information Processing Systems, vol. 31, pp. 5732–5741. Curran Associates, Inc. (2018). http://papers.nips.cc/paper/7815-insights-on-representational-similarity-in-neural-networks-with-canonical-correlation.pdf

19. Moro, N., Dehbaoui, A., Heydemann, K., Robisson, B., Encrenaz, E.: Electromagnetic fault injection: towards a fault model on a 32-bit microcontroller. In: 2013 Workshop on Fault Diagnosis and Tolerance in Cryptography, pp. 77–88. IEEE (2013)
20. 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.) COSADE 2014. LNCS, vol. 9064, pp. 159–173. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-21476-4_11
21. 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), pp. 1106–1111. IEEE (2014)
22. Quisquater, J.J.: Eddy current for magnetic analysis with active sensor. Proc. Esmart **2002**, 185–194 (2002)
23. Raghu, M., Gilmer, J., Yosinski, J., Sohl-Dickstein, J.: SVCCA: singular vector canonical correlation analysis for deep learning dynamics and interpretability. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) Advances in Neural Information Processing Systems, vol. 30, pp. 6076–6085. Curran Associates, Inc. (2017). http://papers.nips.cc/paper/7188-svcca-singular-vector-canonical-correlation-analysis-for-deep-learning-dynamics-and-interpretability.pdf
24. Rohatgi, A.: Webplotdigitizer: Version 4.5 (2021). https://automeris.io/WebPlotDigitizer
25. Schmidt, J.M., Hutter, M.: Optical and EM fault-attacks on CRT-based RSA: Concrete results.na (2007)
26. Skorobogatov, S.: Low temperature data remanence in static RAM. Technical report. UCAM-CL-TR-536, University of Cambridge, Computer Laboratory, June 2002. https://doi.org/10.48456/tr-536
27. 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.) COSADE 2014. LNCS, vol. 9064, pp. 159–173. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-21476-4_11
28. Werner, V., Maingault, L., Potet, M.L.: Fast calibration of fault injection equipment with hyperparameter optimization techniques. In: Grosso, V., Pöppelmann, T. (eds.) Smart Card Research and Advanced Applications. CARDIS 2021. LNCS, vol. 13173, pp. 121–138. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-97348-3_7
29. Wu, L., Ribera, G., Beringuier-Boher, N., Picek, S.: A fast characterization method for semi-invasive fault injection attacks. In: Jarecki, S. (ed.) CT-RSA 2020. LNCS, vol. 12006, pp. 146–170. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-40186-3_8