Glitch it if you can: parameter search strategies for successful fault injection

Rafael Boix Carpi[1], Stjepan Picek[2,3], Lejla Batina[2], Federico Menarini[1], Domagoj Jakobovic[3] and Marin Golub[3]

[1] Riscure BV, The Netherlands,
{BoixCarpi, Menarini}@riscure.com
[2] Radboud University Nijmegen, The Netherlands
{s.picek, lejla}@cs.ru.nl
[3] Faculty of Electrical Engineering and Computing,
Zagreb, Croatia
{domagoj.jakobovic, marin.golub}@fer.hr

Abstract. Fault analysis poses a serious threat to embedded security devices, especially smart cards. In particular, modeling faults and finding effective practical approaches that are also generic is considered to be of interest for smart card industry. In this work we propose a novel methodology to deal with a difficult question of choosing multiple parameters required for effective faults. To this aim, we investigate several algorithms and find a new promising direction using evolutionary computation. Our experimental results on some of the smart cards used today show the potential of this new approach. Our best algorithm is a tailored search strategy especially developed for the purpose of finding the best choice of parameters for glitching. With this approach we found some of off-the-shelf devices, although secured against this type of attacks, still vulnerable.

Keywords: Fault Analysis, Glitches, Smart Cards, Self-Adaptive Algorithms, Evolutionary Computation

1 Introduction

Since smart cards are around in our lives for the past three decades, and becoming ever more pervasive, it seems impossible that we ever lived without them. Yet, at the same time the threats to the security of those small devices are multiple and cheap and at the same time effective countermeasures against various attacks belong to the most extensively researched topics today.

In 1996 Anderson and Kuhn [1] discussed the tamper-resistance of smart cards, and in 1999 Kömmerling and Kuhn presented a set of techniques for tampering with them [2]. It became evident that the possibilities for the adversary are numerous. In general, the techniques for tampering can be classified as *passive* or *active*. In passive techniques some side-channel information is monitored and there is no interference with the normal processing of the card. An example of these passive techniques is the analysis of power consumption, as introduced by Kocher et al. [3] or electromagnetic radiation [4]. In the case of active techniques, the device is not only monitored but also external interferences affect the normal behavior of the device. An example is Fault Injection (FI) and these interferences, the so-called *glitches*, can be of different nature: optical (laser pulses) and electrical glitches (voltage, clock), temperature changes, electromagnetic (EM) radiation, etc. are used to cause some malfunctioning, resulting in some cases in secret key recovery. Fault injection techniques by glitching are typically *non-invasive* techniques, in the sense that the smart card is not physically modified (versus other *invasive* techniques that require hardware modifications).

A fault injection attack is considered to be successful if after exposing the device under attack to a specially crafted external interference, the device shows an unexpected behavior, which can be exploited by an attacker (eg. leaking of sensitive information, bypassing security checks, etc.). However, this external interference has to be precisely tuned for the fault injection to succeed. As an example, a complete characterization of a clock signal glitch requires from the security analyst to define more than 10 parameters (related to clock signal voltage levels, time offset of the glitch, etc.). In addition, hardware designers introduce countermeasures in their devices for preventing fault injection attacks. Hence, finding the correct parameter setting is a highly non-deterministic process, and countermeasures just add up to this non-determinism. As a consequence, security analysts usually set a value range for each parameter, and leave their fault injection setup experimenting over thousands of different parameter configurations within those given ranges to be analyzed off-line afterwards.

Finding the correct parameters for a successful FI can be considered as a search problem where one aims to find, within minimum time, the parameter configuration or ranges of parameter values which result in a successful fault injection. The search space, considering all possible combinations of the values of interest for the fault injection such as voltage, timing, offset, etc., is too large to perform an exhaustive search. For example, there are in total 8 parameters to be set for voltage (VCC) glitching even without considering multiple glitches. As a simple example, testing only 6 values yields $6^8 = 1676916$ parameter combinations! This is unfeasible to test in a reasonable amount of time as it would take over 19 days assuming a quite fast rate of one measurement per second. Here, by a measurement we mean a complete execution of the algorithm of interests on the device including the final response (which can have several different outcomes such as reset, stop, etc.) Considering this problem within the tasks of a security analyst, which often has a very limited or no knowledge about the inner design of the device (*blackbox* testing), setting an accurate range for the parameters can be quite challenging, and a bad estimation of these ranges leads to spending a lot of time in testing parameter combinations that could have been easily discarded upfront.

Due to all these issues and the unfeasibility of performing an exhaustive search due to the time constraints, there is a clear need for a methodology for parameters search that can ultimately lead to a more effective security evaluation. In this work we present several possibilities for finding and tuning the parameters keeping the assumptions on the device under attack as generic as possible. We show several effective approaches that were tested on off-the-shelf devices with different successes. We develop a search strategy that is especially tailored towards a large class of devices of today using common assumptions and defining a new model. Our best algorithm is proven to be efficient against some state-of-the-art protected (against glitching) devices. Furthermore, a new direction based on generic algorithms is also investigated and found suitable when less is known about the device under attack.

1.1 Related Work

The concept of fault analysis-based attacks is known in the research community for around twenty years. Boneh, DeMillo and Lipton published an attack on RSA about exploiting hardware faults for cryptanalysis [5, 6]. The attack described, often also called the Bellcore attack, resulted in numerous contributions in, not just theoretical papers on attacks and countermeasures assuming that faults can be applied, but also in more practical works showing what is really possible in terms of inducing faults. However, the first type of papers are more common, mainly due to the lack of proper equipment at academia. All together, there are only a few works that address the practical issues that arise while applying these techniques.

Kömmerling and Kuhn [2], published a paper in 1999, which is considered to be the milestone in the context of security evaluation against fault attacks. In this work the authors present an extensive collection of techniques for fault injection and other tampering techniques and give hints on how to mitigate some of them. The paper highlights the case of VCC fault injection (referred to as *glitch attacks*) and emphasizes those as the ones most useful in practice.

Aumüller et al. published in 2002 one of the first practical works on fault analysis [7], in which they describe a real-life scenario of the impact of injecting glitches in the VCC and clock lines of an IC. They also suggest some countermeasures applicable in this specific case. Approximately at the same time, Skorobogatov and Anderson introduced optical (laser) fault injection [8], where they describe injecting faults with a laser on a decapsulated IC. This technique is still very successful nowadays for defeating the security of many protected devices, but it is out of scope for this work.

Recent paper from van Woudenberg et al. [9], describes a real attack scenario for an Optical Fault Injection attack. The practical problem of setting the parameters for fault injection is introduced in their work and the authors briefly discuss the lack of methodology to solve it as the main direction they rely on is based on heuristics. In addition, the paper gives a nice overview of all the practical issues that arise during a real execution of the FI attacks on actual hardware. Similarly, the work of Balasch et al. [10] explores the effects of glitches injected in the clock line of an IC. This work is very interesting for identifying various effects that a glitch can cause on real hardware in terms of defining all possible outcomes of a successful fault injection. However, it has to be noted that current smart cards usually run on an internal clock which makes this FI technique unfeasible.

All together, our paper continues this line of research focusing on more practical problems with fault injection but it is also unique. Namely, we first focus on the problem of finding the right set of parameters in order to optimize the glitching effects that can be explored by the adversary. Second, we derive new theoretical framework for this multi-parameters search and apply it on some actual off-the-shelf smart cards. While doing this, we evaluate several search strategies, one of which is using ideas from evolutionary computation. Our contributions are specified more precisely below.

1.2 Our Contribution

Here we summarize the contributions of this work:

- We propose a new methodology to handle the difficult problem of finding the right sets of parameters for glitching. Our methodology is based on a model that is suitable for smart cards of today. Namely, we distinguish between two phases for glitching, one focusing on voltage parameters and the other one on proper timing.
- After experimenting with several approaches, we develop a new search strategy that is time-effective and breaks some off-the-shelf devices.
- We advocate a new direction for this problem building on our first results from the approach based on genetic algorithms.

The remainder of this paper is organized as follows: in Section 2 we give the problem statement and the model we use for the experiments, in Section 3 we present several search strategies and their results. Finally, in Section 4 we conclude the paper and give some suggestions for future work.

2 Problem Statement

The goal is to find a search strategy for VCC FI parameters that lead to a successful fault injection. *Input of the search* consists of the parameters required by the search strategy to proceed, and an estimated initial range for every parameter. *Search space* is a set of all the possible combinations of values for every parameter required to define the VCC FI attack. Parameters that can have real values are considered as discrete-valued parameters sampled with the maximum resolution of the acquisition hardware devices, and all value ranges are bounded. The goal of the search is to get the maximum information about the behaviour of a device with the minimum number of measurements given a black-box scenario. Also, the goal is to find parameters that define a successful VCC FI attack in the case that device is vulnerable to fault attacks caused by glitching. As an *output of the search*, a report of the behavior of device is generated. Additionally, an output can include a parameter combination or a set of parameter combinations that lead to a successful VCC FI attack. Also, a parameter combination or a set of parameter combinations that trigger unexpected behavior of device can be also included although they do not lead to a successful VCC FI attack.

2.1 Model

We divide the search into two phases: in the first phase we look for the appropriate glitch shape (containing all the parameters that define the signal) and in the second phase we look for the timing instant in which we have to inject the fault. The motivation for the parameter split into two stages is obtaining a reduction in the dimensionality (thus, complexity) of the problem. The feasibility of this parameter splitting was experimentally tested to be possible and useful: all TOEs covered by this research (and also TOEs outside the scope of this research) showed a similar behavior w.r.t. glitch shape-related parameters. The second stage search consists of a time sweep with glitch shapes (glitch length, glitch voltage) output by the first stage search. The time range defined in the initial search space is discretized in n time instants[4] . In each time instant, a subset of the glitch shapes output by the first stage is tested.The verdicts of all measurements are reported as the final output of the parameter search. In this paper we give sufficient details for the first search phase only. For the second phase one should proceed similarly.

Two parameters of interest for the first phase are glitch voltage and glitch length. A verdict represents the class that, based on a response from the device, a glitch has been classified to. The assumptions that allow predicting the possible verdict of a measurement given the glitch voltage and the glitch length are as follows:

1. There exists an upper bound for the glitch voltage, VLOW[5], and if the glitch voltage is set to this value or higher, device will just ignore the glitch (it will interpret it as signal noise), and a NORMAL verdict will be obtained.
2. There exists a lower bound for the glitch voltage, VHIGH[5], and if the glitch voltage is set to this value or lower, device interprets the glitch as a power cut or as an attempt to tamper with it, and a RESET or MUTE verdict will be obtained.
3. There exists a lower bound for the glitch length, LLOW, and if the glitch length is set to this value or lower, device will just ignore the glitch, and a NORMAL verdict will be obtained.
4. There exists an upper bound for the glitch length, LHIGH, and if the glitch length is set to this value or higher, device interprets the glitch as a power

[4] In the time dimension, the response of the TOE could be different each time instant. However, due to the presence of internal unstable clocks in TOEs Target B and Target C, the glitch offset has been omitted in the search. The clock jitter causes a FI time instant spread bigger than the accuracy we can obtain with the testing equipment by setting a precise glitch offset in time (2 *ns). Additionally, the model assumes a stable operation of the TOE, and not a drastically changing power profile over time (e.g. TOE booting) for the validity of glitch shape-related parameters in the 2nd stage of the search.*

[5] Note that small glitches that are to be ignored have a length close to LLOW and voltage close to VLOW, but the glitch voltage is typically a negative value, hence the counter-intuitive naming convention for voltage boundaries.

failure or as an attempt to tamper with it, and a RESET or MUTE verdict will be obtained.

5. If the glitch voltage and the glitch length take values in the ranges of (VLOW, VHIGH) and (LLOW, LHIGH) respectively, the response of device depends also on the rest of the parameters of the glitch (both from the glitch shape and the glitch timing).

The explanations for the possible verdict classes are given below.

Verdict from the class NORMAL will be obtained if the device response was expected, verdicts RESET and MUTE are derived if the device responds accordingly while performing a measurement. If the device is vulnerable to FI, the verdicts from the class INTERESTING can be found. It points to the area defined by the decision boundary between the plane regions corresponding to the NORMAL and RESET/MUTE regions plus some threshold distance. If these two regions overlap, the class INTERESTING is to be found in the intersection of these two plane regions. (We assume here a two-dimensional space with only the glitch length and glitch voltage parameters.) The verdict CHANGING is found in the same area as the INTERESTING verdict. This verdict class is assigned when two measurements with the same parameter configuration for the glitch shape yield different verdicts. The verdict SUCCESSFUL is to be found inside the (glitch voltage, glitch length) area which produces the INTERESTING verdicts where can be more than one combination of parameters that yields a SUCCESSFUL verdict.

3 Experiments and Discussion

In this section we present different search strategies and their experimental results on several smart cards. First we give additional information about search space settings followed by the experiments. Afterwards, we present a comparison among different search strategies in terms of their effectiveness.

3.1 Search Space Settings and Experiment Definition

The initial search space parameters are given in Table 1.

The experiments are performed as follows:
For each tested device, several runs of each strategy for the first stage of the search are executed. Besides the common parameters already mentioned in Table 1, we also use the following algorithms:

- MonteCarlo search (baseline): 2048 measurements
- FastBoxing: 2 iterations (*maxIter*=2), $4 \cdot 4 \cdot 64$=1024 measurements per iteration (n=4, *numMeas*=64), 10 000 maximum iterations (*maxIter*=10 000)

[6] The number of glitches was chosen as a random value due to not observing any statistically significant change in the TOE response w.r.t. this parameter within the given range.

Table 1. Search Space parameters

Parameter name	Parameter value
Glitch voltage	[-5, -0.05] V
Glitch length	[2, 150] ns
VCC Voltage VCC	5 V
CLK High Voltage	5 V
CLK Low Voltage	0 V
CLK signal frequency	1 MHz
Number of glitches	random value from [1, 10] [6]

– Adaptive zoom&bound: 10 000 maximum iterations (*maxIter*=10 000), $4 \cdot 4$ grid (n=4), 1 and 3 measurements per iteration (*numMeas*=1, *numMeas*=3)
– Genetic Algorithm: maximum number of generations = 20, population size=30, maximum number of consecutive generations without improvement=50

3.2 Experimental results

The tests are conducted on three targets. Target A is unprotected smart card and is therefore suitable for the training phase. Smart cards B and C are protected against several FI techniques, especially VCC FI. Since one of the possible outcomes of a VCC FI attack is permanent malfunction of a device, multiple samples of each card were used. For all search strategies, all samples from the same device showed the same physical behavior w.r.t. glitch shape-related parameters. In this sense, the glitch shape parameters found for a device sample are valid for all samples[7] of the same device. This behavior was not observed for the time-related parameters.

For the table listings, the following abbreviations are used:

– *TestReps*: number of repetitions of the test
– *MeasInTest*: average total number of measurements in tests, if MeasInTestT then it includes first and the second stage.
– *INT(M)*: number of measurements with a INTERESTING verdict class. The figure is presented as the median value of all values in the tests. The choice of the median is for reflecting the typical performance of the search strategy.
– *INT(%)*: number of measurements with a INTERESTING verdict class per hundred (%). This value is computed from the sum of all INTERESTING measurements in all tests divided by the accumulated number of measurements in all tests, and normalized to 100 measurements.
– *SUC(M)*: same as INT(M) but for the SUCCESSFUL verdict class.
– *SUC(%)*: same as INT(%) but for the SUCCESSFUL verdict class.

[7] For each device all samples were from the same batch, hardware revision and manufacturer.

3.3 Monte Carlo Strategy and Results

This search strategy consists of performing measurements with randomly selected parameter combinations within the given initial search space. The random distribution for selecting values is considered to be uniform for each parameter present in the search space. This search strategy is considered as the baseline search strategy. The short test runs with 3072 measurements had no SUCCESSFUL measurement, and only the 76800 measurement test run produced 11 SUCCESSFUL measurements.

Furthermore, due to the random nature of the parameter selection there is a significant number of repeated parameter combinations (glitch length, glitch voltage) for the glitch shape. This repetition is interesting if it is made in the plane region that yields the INTERESTING verdict class. However, it is highly undesired for measurements in which the device response is predictable.

3.4 FastBoxing Strategy and Results

FastBoxing algorithm is a simple, iterative algorithm devised for the automatic setting of the parameters in the first stage of the search. The algorithm works in the following way: search strategy assumes the boundaries for the glitch shape: VHIGH, VLOW and LLOW, LHIGH. The search algorithm will try to find these boundaries by doing two steps: a *measurements step* and a *reflection step*. For each iteration, the measurement step consists of a sampling of the search space and then it performs measurements at the sampled points. After performing the measurements, the algorithm will start the reflection step for finding out an estimate of the VHIGH, VLOW and LLOW, LHIGH boundaries. For the VLOW boundary, all points to its right should produce the NORMAL verdict, so the algorithm does the following. First column of points starting from the right is analyzed. If all the points of this column belong to the NORMAL verdict class, then the next column to its left is analyzed. If all points in the second column also belong to the NORMAL verdict class, the algorithm estimates that the VLOW boundary is not between those two columns. When the algorithm finds a column that contains some points belonging to NORMAL verdict and some points belonging to RESET or MUTE classes, it estimates that the VLOW boundary is between that column and the column to its right. Once this estimation has been done, the VLOW boundary is temporarily set to this estimation. For the rest of the boundaries, the process is analogous.

Once the algorithm stops, the last estimation for the boundary values for the glitch length and glitch voltage will be the output. The next search stage will sample points inside the box bounded by the VHIGH, VLOW, LLOW and LHIGH boundary values for its input.

In Table 2 we summarize the results of the tests for the FastBoxing strategy. The second stage of the parameter search is performed with a set of 10 (glitch length, glitch voltage) parameter combinations randomly selected from the bounded region in the OUTPUT of the FastBoxing search strategy.

Table 2. Results for the FastBoxing search strategy

Device	TestReps	MeasInTestT	INT(M)	SUC(M)	INT(%)	SUC(%)
Target A	5	3048 (2048+1000)	26	9	0.800	0.361
Target B	5	3048 (2048+1000)	0	0	0.00	0.00
Target C	1	3048 (2048+1000)	0	0	0.00	0.00

In the case of the FastBoxing search for the vulnerable device Target A, the inaccurate estimation of the INTERESTING verdict class results in poor (glitch length, glitch voltage) parameter combination choices. This is especially noticeable if these parameter combinations are close to the boundary values. Because of this, the number of SUCCESSFUL measurements varies significantly depending on the random selection of parameter combinations. As an example, Run 2 of the test yielded 26 SUCCESSFUL measurements, whereas Run 3 of the test yielded only 4 SUCCESSFUL measurements.

It is worth mentioning that, to focus on the search space region in which the INTERESTING verdict class is found, the performance of the search improved significantly. All test runs of the FastBoxing search strategy yielded INTERESTING and SUCCESSFUL measurements.

3.5 Adaptive zoom&bound Strategy and Results

The Adaptive zoom&bound search strategy iteratively bounds the region that yields the INTERESTING or CHANGING verdict classes and "zooms" inside that bounded region. This is achieved by decreasing the distance between new measurements in the glitch shape search space. The region bounding is performed in an adaptive way, similar to a 2D version of a binary search. The Adaptive zoom&bound search uses the same two-step iterative process as FastBoxing algorithm, but the processing done in the reflection step is different. Reflection step works as follows: the distance between two neighbour points is set to a *pointDist* variable. The measurements are placed in a 2D plane for the glitch shape (just for ordering them). The horizontal axis is the glitch voltage parameter, and the vertical axis is the glitch length parameter. For each one of the available measurements from the last test, all neighbours of a measurement are checked for their verdict class. If all neighbour measurements in the 2D plane belong to the same verdict, the decision boundary between verdict classes is not found between the point and its neighbours. If a neighbour measurement in the 2D plane belongs to a different verdict class then the boundary is estimated to be between them. A new measurement is added for the test in the next iteration and placed at a distance $pointDist/2$ between them. When all points have been analysed, a new test has been generated with a list of measurements only in the estimated region that produces the INTERESTING verdict class. The algorithm stops if all measurements in the initial measurement step belong to the same class, if no new test measurements are generated during the reflection step, or when the distance between neighbour points has reached the maximum resolution of the hardware devices. Once the algorithm stops, it outputs the set of glitch shape parameters that bound the region producing the INTERESTING verdict class.

The output of the algorithm is a set of the glitch shape parameters (glitch length, glitch voltage) of the measurements that are considered to be bounding the region that yields the INTERESTING verdict class. The output also contains the glitch shape parameters of the measurements with INTERESTING, CHANGING and SUCCESSFUL (if any) verdict classes. The decision of which glitch shapes should be the output is implemented by taking the measurements produced in the last iteration of the algorithm.

In Table 3 we summarize the results for performed tests for the Adaptive zoom&bound strategy.

Table 3. Results for the Adaptive zoom&bound search strategy, 1 performed VCC FI attack per measurement

Device	TestReps	MeasInTestT	INT(M)	SUC(M)	INT(%)	SUC(%)
Target A	5	1198 (**198**+1000)	47	13	**3.895**	**1.064**
Target B	5	1128 (**128**+1000)	0	0	0.00	0.00

The results of the Adaptive zoom&bound strategy are better than in previous search strategies. In particular, the number of measurements required for completing the first stage of the parameter search is very low, so the search speed is improved significantly. For the initial search space used throughout the experiment, the optimum performance is computed as follows:

$$N = n \cdot \lceil max(log_2(\text{rangeV}/\text{resolutionV}),$$
$$log_2(\text{rangeL}/\text{resolutionL}))\rceil = (4 \cdot 4) \cdot \lceil max(log_2(5/0.05), log_2(150/2))\rceil = 112\,\text{meas.}$$

In the case of Target A, the search strategy has more measurements due to the device behavior in the search space region that produces the INTERESTING verdict class. Additionally, the number of INTERESTING and SUCCESSFUL measurements are almost four times larger than those for the FastBoxing algorithm.

Figure 1 shows the plot for the first stage of the parameter search in the case of the Adaptive zoom&bound strategy.

It can be seen that most of the measurements are performed near the decision boundary between verdict classes. Also, the distance between measurements w.r.t. glitch shape parameters is very small. This allows to bound the region producing the INTERESTING verdict class quite accurately. The Adaptive zoom&bound search strategy also allowed to experimentally observe that

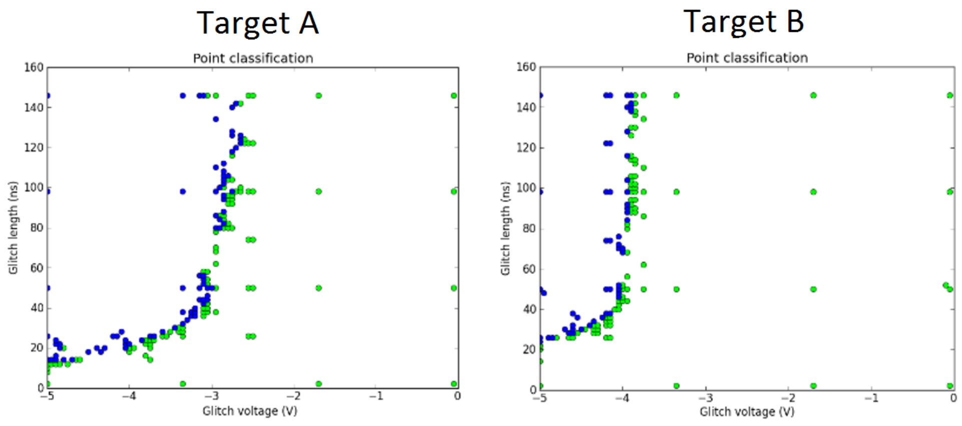

Fig. 1. First stage plot of explored measurements for different devices by the Adaptive zoom&bound search strategy. Green is NORMAL, blue is RESET (A) or MUTE (B)

the glitch shape parameters (glitch length, glitch voltage) are the same for different samples of the same device. Figure 2 shows the plot of the measurement classification for the first stage parameter search of two Target A samples. It can be experimentally verified that a (glitch length, glitch voltage) glitch shape parameter value that leads to a SUCCESSFUL verdict in one sample also does the same in other targets. It can be said then that the glitch shape parameters are exportable between samples of the devices. In contrast, time-related parameters that produce SUCCESSFUL verdicts do change between samples.

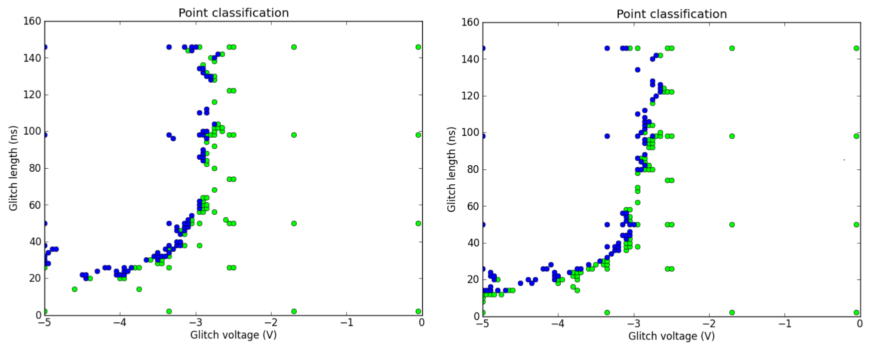

Fig. 2. First stage plot of two samples of Target A.

Here we also present a successful VCC FI attack on Target C. This target incorporates specific countermeasures against VCC fault injection, as indicated by the manufacturer. In addition, this card has been granted the EAL4+ certification level of Common Criteria. This means that the device has been previously tested by an independent security evaluation lab against different attack techniques, including VCC fault injection. The output of the first stage of the search is depicted in Figure 3.

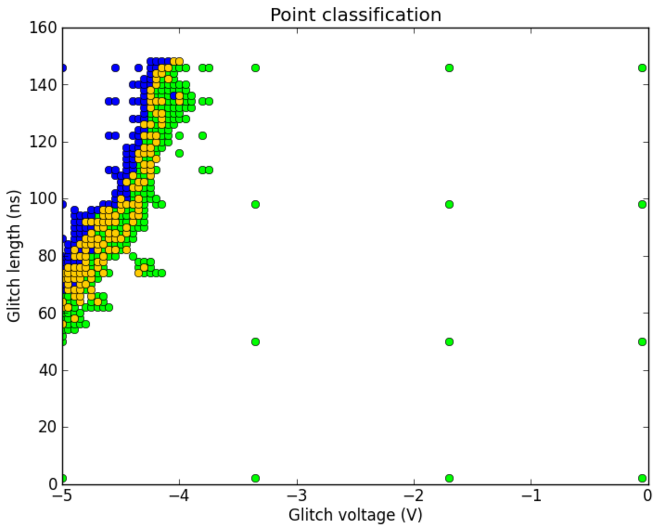

Fig. 3. Output of the measurement classification for Target C by the Adaptive zoom&bound search strategy, 3 repetitions per measurement. The orange color depicts the CHANGING verdict class.

We can see that due to the focus in the search space region producing the CHANGING verdict class and the multiple attempts per measurement, the jitter was mitigated and suitable (glitch length, glitch voltage) parameter settings could be found. Table 4 shows the results for the performed parameter search. As far as the authors know, this target was not known to be vulnerable to VCC FI attack before.

Table 4. Results for the Adaptive zoom&bound search strategy with Target C, 3 VCC FI attacks performed per measurement

Device	Measurements1ststage	Measurements2ndstage	INT(M)	SUC(M)
Target C	812	1000	17	**19**

3.6 Genetic Algorithm Strategy and Results

Besides using deterministic algorithms as the two examples mentioned above, it is also possible to use heuristic algorithms. Since finding the correct parameter setting is a non-deterministic process that can be considered as a search problem, it is natural to try to use non-deterministic algorithms. Genetic algorithms are a subclass of evolutionary algorithms where the elements of the search space S are arrays of elementary types [11]. Since Genetic Algorithms (GAs) are typically used as function optimization algorithms, a fitness function must be defined for mapping the different verdict classes present in the device model to the fitness values. In particular, the mapping currently used is: NORMAL verdict class has value 1, RESET or MUTE verdict classes have value 2, INTERESTING verdict class has value 8, CHANGING verdict class has value 8.5 and SUCCESSFUL verdict class has value 10. Formally, the GA aims to find a (glitch length, glitch voltage) tuple such that the fitness value F is maximal. To be able to use GA on this problem, a generic GA is modified and instead of the standard operators we use custom selection and crossover operators. The GA generates an initial population of n random combinations of (glitch length, glitch voltage) parameter values. Each individual of each generation is assigned its corresponding fitness value. Each population is evolved into a new generation of the population by means of an evolution step (iteration step). The evolution step performs the following tasks: in the crossover, GA takes two individuals from different verdict classes and produces a new individual with a (glitch length, glitch voltage) parameter configuration between the values of the two parent individuals. To perform the mutation step, some individuals evolve by adding to their parameter values a random value. Finally, the algorithm preserves a certain number of individuals with the highest fitness value in the next generation.

GA performs evolution steps until a maximum number of evolution steps is reached, or until a specified number of generations without improvement is reached. A modification that has been introduced into the GA is the notion of a "good enough" fitness value. The algorithm has an internal fitness threshold value, and all generated individuals that have a fitness value equal or higher than the threshold value will be output by the algorithm as the OUTPUT of the first stage of the parameter search. With the current fitness function definition, a threshold value of 8 outputs all the measurements that had an INTERESTING, CHANGING or SUCCESSFUL verdict class. For evolutionary algorithms test suite we use the Evolutionary Computation Framework (ECF) [12]. ECF is a C++ framework intended for the application of any type of the evolutionary computation, developed at the University of Zagreb.

3.7 Comparison among different search strategies

In order to have an overview of the performances of the presented search strategies, Tables 5 and 6 contain the best observed metrics in tests performed with Target A (vulnerable to VCC FI) and Target C (presumably not vulnerable to VCC FI). The configuration of the second search stage is the same for all search

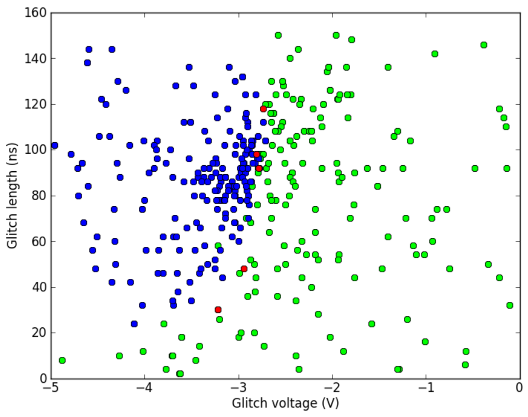

Fig. 4. First stage plot for the measurements of the GA with the Target A.

strategies (except for the Monte Carlo search): 10 (glitch voltage, glitch length) parameter combinations, 100 time instants.

For the table listings, the following abbreviations are used:
Meas 1st Stage: total number of measurements in 1st stage of the parameter search;
SUC: ratio of SUCCESSFUL measurements versus the total number of measurements (1st+2nd stages), normalized to 1/100;
Total_I: total number of INTERESTING measurements;
Total_S: total number of SUCCESSFUL measurements;
Total_M: total number of measurements (1st+2nd stages).

Table 5. Metrics of the different search strategies for Target A.

Strategy	Meas 1st Stage	SUC(%)	Total_I	Total_S	Total_M
Monte Carlo	N/A	0.0000	19	0	3072
FastBoxing	2048	0.29526	26	9	3048
AdaptZoom	**192**	**1.17450**	**56**	**14**	**1192**
GA	1560	0.3125	21	8	2560

Looking at the results, the best overall strategy is the Adaptive zoom&bound search strategy. It completes the first stage of the search with the least number of measurements and it has the best ratio of SUCCESSFUL measurements, and produces the most INTERESTING and SUCCESSFUL measurements. The use of information available allows to quickly direct the parameter search towards the most promising parameter configurations.

The GA shows a promising performance, because it is able to produce a significant number of INTERESTING and SUCCESSFUL measurements. However, the performance in terms of number of measurements in the 1st stage is not very good. This is due to a large number of generations being produced without significant improvements. A parameter tuning on this approach should result in a better performance of the algorithm. This parameter tuning, in combination with the addition of new features to the algorithm, is left as future work.

Table 6. Metrics of the different search strategies for Target B.

Strategy	Meas 1st Stage	SUC(%)	Total_I	Total_S	Total_M
Monte Carlo	N/A	0	0	0	3072
FastBoxing	2048	0	0	0	3048
AdaptZoom	128	0	0	0	1128
GA	6868	0	0	0	7868

Finally, it should be mentioned that the Monte Carlo search strategy has been found to be the most inefficient search strategy. However, due to its random nature, it is still a viable option if no restriction on the number of measurements is imposed.

4 Conclusions and Future Work

This work deals with the so-far unexplored topic of finding the right parameters for successful faults by glitching. We experiment with several search strategies and find some promising methodologies that are effective against some proprietary smart cards with glitching countermeasures. The best method is rather generic and shows good results against different devices. Finally, we identify another promising direction using genetic algorithms that can be further optimized as future work.

Acknowledgements

This work was supported in part by the Technology Foundation STW (project 12624 - SIDES), The Netherlands Organization for Scientific Research NWO (project ProFIL 628.001.007) and the ICT COST action IC1204 TRUDEVICE.

References

1. Anderson, R., Kuhn, M., A, E.U.S.: Tamper resistance — a cautionary note. In: In Proceedings of the Second Usenix Workshop on Electronic Commerce. (1996) 1–11
2. 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. WOST'99, Berkeley, CA, USA, USENIX Association (1999) 2–2
3. Kocher, P.C., Jaffe, J., Jun, B.: Differential power analysis. In: Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology. CRYPTO '99, London, UK, UK, Springer-Verlag (1999) 388–397
4. Quisquater, J.J., Samyde, D.: ElectroMagnetic Analysis (EMA): Measures and Counter-Measures for Smart Cards. In: Proceedings of the International Conference on Research in Smart Cards: Smart Card Programming and Security. E-SMART '01, London, UK, UK, Springer-Verlag (2001) 200–210
5. D. Boneh, R.D., Lipton, R.: New threat model breaks crypto codes. Bellcore 85 Press Release (1996)
6. Boneh, D., Demillo, R.A., Lipton, R.J.: On the importance of checking cryptographic protocols for faults, Springer-Verlag (1997) 37–51
7. Aumüller, C., Bier, P., Fischer, W., Hofreiter, P., Seifert, J.P.: Fault attacks on RSA with CRT: Concrete results and practical countermeasures. In: Revised Papers from the 4th International Workshop on Cryptographic Hardware and Embedded Systems. CHES '02, London, UK, UK, Springer-Verlag (2003) 260–275
8. Skorobogatov, S.P., Anderson, R.J.: Optical fault induction attacks. In: Revised Papers from the 4th International Workshop on Cryptographic Hardware and Embedded Systems. CHES '02, London, UK, UK, Springer-Verlag (2003) 2–12
9. van Woudenberg, J., Witteman, M., Menarini, F.: Practical optical fault injection on secure microcontrollers. In: Fault Diagnosis and Tolerance in Cryptography (FDTC), 2011 Workshop on. (2011) 91–99
10. Balasch, J., Gierlichs, B., Verbauwhede, I.: An In-depth and Black-box Characterization of the Effects of Clock Glitches on 8-bit MCUs. In: Proceedings of the 2011 Workshop on Fault Diagnosis and Tolerance in Cryptography. FDTC '11, Washington, DC, USA, IEEE Computer Society (2011) 105–114
11. Weise, T. In: Global Optimization Algorithms Theory and Application. (2009) http://www.it-weise.de/.
12. Jakobovic, D., et al.: Evolutionary computation framework (January 2013) http://gp.zemris.fer.hr/ecf/.

5 Appendix: TOE details

A more detailed description of the TOEs described in this paper follows:
Target A: It is a smartcard based on an ATMega163+24C256 IC, CMOS technology, hardware last revision 2003. This TOE does not have any side-channel countermeasure nor fault-injection countermeasure. All processing of the card is performed in software, and the card was running on an external 4MHz clock frequency. In particular, this target is also available from Riscure BV as the research target "Training Card 6". The code that was attacked was a vulnerable PIN (Personal Identification Number) authentication mechanism is as follows:

```
...
for (i=0;i<4;i++) {
    if (pin[i] == input[i])
        digits_ok++;
}
    if (digits_ok==4)  //BRANCH STATEMENT == CODE UNDER ATTACK
        respond_code(0x00,SW_NO_ERROR_msb,SW_NO_ERROR_lsb);
    else
        respond_code(0x00,0x69,0x85);
...
```

Target B: It is a smartcard bought in 2013 from a webshop from one of the leading manufacturers in the sector. This TOE is a protected target, and has countermeasures against SCA and FI, such as fault injection detection logic and light, temperature and clock sensors. The IC design is from late 2004. Additionally, it has dedicated logic for cryptograpic operations. More in detail, this TOE implements the JavaCard OS 2.2.1 and GlobalPlatform 2.1.1 standard. It runs on an internal, unstable clock at an unknown frequency. The supplied external clock frequency was 4MHz. The card was running exclusively on software (no crypto hardware present in the IC was used). The Java applet loaded into the card was a double nested loop with two counters and a checksum. The code was similar to the following piece of code:

```
...
for(outerLoopCounter=0;outerLoopCounter<2;outerLoopCounter++){
    checkpoint=1;
    for(innerLoopCounter=0;innerLoopCounter<1000;innerLoopCounter++){
        checkpoint=2;
        dummyOperation1();
        iterations=iterations+1;
    }
    checkpoint=3;
    dummyOperation2();
}
sendBytesToTerminal(outerLoopCounter,innerLoopCounter,iterations,
valueFlag);
...
```

Target C: It is a smartcard bought in 2013 from a webshop from one of the leading manufacturers in the sector. This TOE is a protected target, and has the same feature set as *Target B* in terms of hardware and countermeasures. This TOE implements the JavaCard OS 2.2.1 and GlobalPlatform 2.1 standard. It was also Common Criteria certified level EAL4+ in 2008. The Java applet loaded into the card was the same applet as described for *Target B*.