Multi-source Fault Injection Detection Using Machine Learning and Sensor Fusion

Ritu-Ranjan Shrivastwa, Sylvain Guilley, Jean-Luc Danger

Multi-Source Fault Injection Detection using Machine Learning and Sensor Fusion

Ritu-Ranjan Shrivastwa [1,2] [0000−0002−8909−0406],
Sylvain Guilley [1,2] [0000−0002−5044−3534], and
Jean-Luc Danger [2] [0000−0001−5063−7964]

[1] Secure-IC S.A.S., Rennes, FRANCE
{ritu-ranjan.shrivastwa, sylvain.guilley}@secure-ic.com
[2] LTCI, Télécom Paris, Institut Polytechnique de Paris, Paris, FRANCE
{ritu.shrivastwa, sylvain.guilley, jean-luc.danger}@telecom-paris.fr

Abstract. Fault attacks have raised serious concern with the growing amount of connected devices. Even a small vulnerability might compromise a complete network. It is therefore important to secure all the devices in the connected architecture. A solution to this problem is presented in this paper where we provide a hardware framework, called Smart Monitor, that utilizes a set of sensors (digital or physical) placed on the chip alongside the security target to be protected. The framework continuously monitors the status of the sensors and its Artificial Intelligence (AI) core produces two outputs viz. presence of a fault and type of detected perturbation. The types of attack sources can be electromagnetic, clock-glitch, laser, temperature, etc. In this work we utilize Electro-Magnetic (EM) and Clock-Glitch (CG) as sources of fault injections. Both attacks are performed in multiple settings to increase attack diversity. The framework is able to detect the presence of an attack with 92% accuracy for mixed or multiple attack sources, and further classify the type of perturbation with 78% accuracy keeping the false positive rate at 0%. Overall, this two-stage detection framework is a cost-effective countermeasure that can be deployed easily in any integrated circuit to safeguard against multiple fault attacks. The AI core is further evaluated for consistency in performance on hardware using High Level Synthesis (HLS), as a proof-of-concept, emulating real-world scenario.

Keywords: Internet of Things (IoT) · Artificial Intelligence (AI) · Machine Learning (ML) · Threat Detection · Cyber-Protection · Decision Making Process · Naive Bayes Classifier · Embedded Security · Cyber-Physical Attacks · High-Level Synthesis (HLS).

1 Introduction

1.1 Motivation

The growth rate of connected devices is on a fast pace and soon the Internet-of-things might be as huge as the Internet itself. This draws massive attention,

in terms of security, towards all the devices that form the nodes of this network including the end-devices. The breach can be made through these nodes via active or passive attacks [26]. Passive attacks include Side-Channel Attacks (SCA) that is well researched, to break even the complex cryptosystems, where mostly the device vulnerabilities are exploited to obtain the security parameters [24]. Active (fault) injections like laser, EM, CG, etc. can be performed to disrupt the normal functions of the device and eventually forcing the system to malfunction in order to bypass security [8, 11]. Precise attacks can be made to execute targeted operations inside the chips, like skip or replace instructions, to perform undesired operations [9]. In this paper we focus on the active attacks and provide a countermeasure that is able to detect multiple such attacks. In other words, a single solution to detect fault injection attacks from multiple sources on the same target.

1.2 Our contribution

We present a two-stage detection framework to detect active fault attacks. This Machine Learning (ML) based system is able to detect the presence of multiple types of attack. The first stage reports the presence of an attack and the next stage predicts its type. ElectroMagnetic Fault Injection (EMFI) and Clock-Glitch Fault Injection (CGFI) are performed on a chip that runs a standard encryption algorithm. The same chip is also mounted with precise digital sensors (DS) that are the core components used for the detection. The ML based controller, called Smart Monitor, continuously monitors the statuses of the sensors and generates two outputs viz. attack status and perturbation type. In this work we utilize two attack sources, however, other sources can be added to the same framework. We validate the design using HLS by performing benchmark testing with the test vectors used to derive the classification results at the software level.

The rest of the paper is organized as follows: Section II provides a background of Fault Injection attacks and Machine Learning, Section III presents the entire framework structure and HLS evaluation with hardware performance and utilization, Section IV provides the results of various stages of detection, and conclusion in Section V.

2 Background

2.1 Fault Injection attacks

Fault Injection Attacks (FIA) have been around for quite some time and have been extensively used to cause glitches in the Integrated Circuits (IC) to eventually extract the secret parameters or cause malfunction to the system. One of the early fault analysis was done in [3]. Several vulnerabilities exposed due to fault attacks have been reported along with their countermeasures like in [2] and [4]. A new class of fault attack was introduced in 2017 by Tang et. al. in [23], known as

CLKSCREW, where they exposed the vulnerabilities of the energy management mechanisms, basically the Dynamic Voltage and Frequency Scaling (DVFS), to break security without the need for physical access to the devices or any equipment to inject fault by overclocking at the software level in ARM processors thereby compromising the Trusted Execution Environment (TEE). More recent attacks, targeting the DVFS, have been proposed such as Plundervolt [15] and VoltJockey [17]. In [7] the authors illustrate the use of Laser fault injection ease the process of syndrome decoding of code-based public-key cryptosystems. A background on Electromagnetic and Clock-Glitch fault injections, considered for this work, is detailed as follows.

Electromagnetic based EMFI has several implications in retrieving secret data using fault analysis. There have also been works to tamper the control flow of a program by causing instructions skip [18] with high precision. Another detailed study of EMFI impact on Instruction Set Architecture (ISA) was done in [16]. A recent work on the effect of data transfer due to EMFI was done in [12] with a byte-level precision.

Clock-Glitch based CGFI is relatively easier to perform as compared to EMFI. The technique can be in temporarily increasing the clock frequency to either cause some flip-flops to sample their inputs before the new state is reached [1] or reduce the processor's time to write a jump address and prevent the branch execution [14].

2.2 Detecting Fault attacks with Machine Learning

The use of Machine Learning (ML) in the security domain is relatively new. Major works have been done in the Side Channel Analysis [10, 22, 25]. In [20], the authors provide a ML-assisted technique to explore and characterize the fault attack space and use the knowledge of a known fault attack on a cipher in understanding new attack instances. Regarding detection of fault attacks, a recent work [19] is presented that evaluates fault induced leakages from non-cryptographic peripheral components of a security module, targeting cipher implementations, using a Deep Neural Network (DNN) test. From the defensive side, while most fault analyses are based on the characterized faults from known attacks (like [6]), our work is based on a completely different approach of online identification of attacks using real-time sensors. To the best of the knowledge of the authors, our work is the premiere in providing a hardware based framework to dynamically detect FIA from multiple sources.

3 Proposed Methodology and Design Idea

3.1 Digital Sensor

Digital Sensor (DS) is a light weight delay chain unit that can be placed anywhere on the chip fabric. The DSs have delay chains longer than the critical path and

thereby detecting delay faults before they affect the user logic [21] (Fig. 14, page 189). The Digital Sensor is designed to detect various FIAs, such as clock glitch, power glitch, underfeeding, heating, laser attack and electromagnetic. A DS converts all observed stresses into a timing stress for measurement. It is extremely sensitive to variations in temperature and voltage as well as to internal activities of the Device-Under-Test (DUT) which makes it a generic sensor that can detect multiple perturbation types. For a UMC design kit with 28nm HPC (High Performance Computing), with a frequency of 100MHz, each Digital Sensor is 2.93 kGE (kilo Gate Equivalent), which means 730µm^2 for this technology node.

3.2 Smart Monitor

Smart Monitor (SM) is a customizable framework that utilizes a fleet of DSs (aggregating their values) along with the capability of having other physical precision sensors attached to it to analyze and synthesize appropriate outcome. The core of the SM is an AI engine that accumulates all the statuses and evaluates them to produce a meaningful response. We call this method of using all sensor data Sensor Aggregation strategy (see Figure 1).

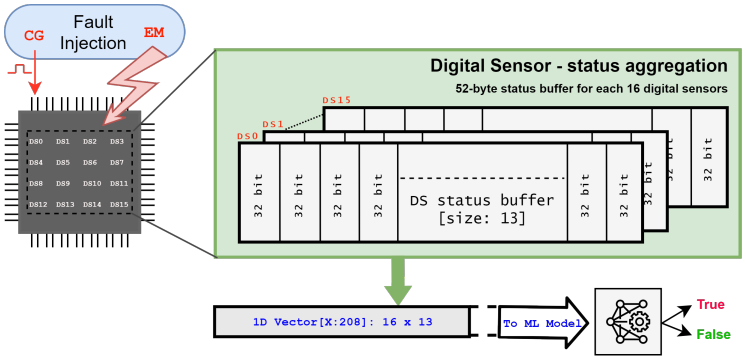

Fig. 1: Illustration of data acquisition from multiple sensors (sensor aggregation) for EM and CG fault injections and their formatting for ML algorithms

Sensor aggregation is important to prevent single DS saturation, a condition in which all the status bits are true/high, which cannot determine the difference between the sources of attacks i.e. the perturbation type, since there is no resolution left to differentiate. Multiple sensors placed at different locations on the chip can have varied impacts based on the type of perturbation and, thus, produce different statuses that can contain information (features) of the attack type and can be easily learnt by the AI engine. This helps to classify among the types of attack using a ML classifier.

3.3 Dataset Information

The EM and CG fault injections are recorded with sixteen Digital Sensors placed inside a Sakura-G FPGA (Field Programmable Gate Array) board, running a standard AES algorithm. In each of the sessions for EM and CG data recording, for every setting, traces have been recorded for both the nominal (without injection) condition and the injected (with fault injection) condition.

A status buffer is associated with each DS. The buffer can store thirteen sensor values and works as a shifting queue i.e., every time (N cycles, depending upon the implementation) a new status is pushed on the top of the stack, the bottom status pops out. The EMFI is performed at four random locations on the chip surface.

A similar session has been carried for the CG fault injection trace acquisition with different CG parameters with varying delay between 379 nanoseconds (ns) to 511 ns with a pulse width of 5 ns, 8 ns and 1.5 ns.

3.4 Machine Learning based evaluation using Two-Stage Detection Framework

Recall the motive of the evaluation, which is to identify if there is any fault injection being performed on the DUT. Conventionally, a single sensor (physical or digital) is enough to detect any kind of unusual activity inside the chip like in [5] and [13] and produce acceptable accuracy. However, most works have been conducted on single type of Fault Injection scenarios (like EMFI) and the detection algorithm relies upon statistical and mathematical models. In this work, we propose a generic framework that is capable of taking multiple types of fault injection scenarios as inputs and deduce whether the DUT is in nominal condition or it is experiencing a fault injection attack attempt. Since the problem reduces to classification of the state of the DUT, it can easily be modelled on a ML classification algorithm. The core engine of the framework is, thus, based on ML algorithms that are trained with example scenarios of nominal and attack conditions.

Another extended aspect of the proposed framework is in its capability of predicting the type of attack. The modalities of operation of the two-stage detection framework can be understood from the simple illustration in Figure 2. The first stage is responsible for detecting the presence of an attack and the second stage is responsible for detecting the type of the attack. This splitting of the functionality in two stages keeps the ML models in binary detection mode[3] which in turn speeds up the process of injection detection to make the system more responsive and also gives the user flexibility to disregard the second stage if not required.

[3] The output classes in second stage will be same as the expected number of attack sources i.e. multi-class classification scenario. However, in our case since we are utilizing only EMFI and CGFI sources, the second stage can also be served by a binary classifier.

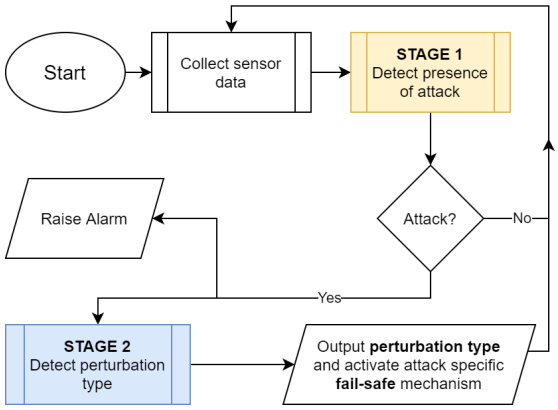

Fig. 2: Proposed framework's modalities of operation. A high level control and data flow diagram of the multi-sourced attack detection with two-stage detector results.

ML Classification Algorithms The choice of ML classifiers is kept to a minimalistic model to enhance latency of detection while maintaining lower footprint on the silicon fabric for the interests of space and power. The popular binary classifier for Gaussian data is the Gaussian Naive Bayes Classifier (GNBC). Given that the sensor data has limited features and does not require non-linear ML algorithms to form a separation, we proceed with the idea that linear models such as GNBC and Logistic regression classifier (LRC) might be able to produce acceptable accuracy. To verify this idea, we perform classification over two linear and two non-linear ML models. Among the non-linear models we choose the popular Support Vector Machines (SVM) and a simple Multi-Layered Perceptron (MLP) neural network.

Modes of evaluation of FIA To describe the detection capabilities (with scores) of the ML models with the EM and CG fault injection datasets, multiple modes of evaluation are performed. Primarily, the classifier should be able to differentiate between nominal and injection classes (labelled as: 0 and 1 respectively). Additionally, the model should be able to detect the source of attack based on selected features from the training datasets. Since the sensors currently used have status saturation directly proportional to the strength of the attack, the problem of classifying the type of attack becomes harder. This is due to the fact that at highest saturation levels of the sensors, it becomes difficult to differentiate between the types of attack. This is why we resort to the sensor-aggregation strategy where multiple sensors are placed at different locations and, even though some sensors might be saturated, some sensors might not be. However, there may be a case where even all the sensors get fully saturated, diminishing the class difference boundary of the attack sources which results in predicting wrong classes.

Therefore, we propose the sensor aggregation strategy where we place multiple (in our case 16) sensors across the entire chip. The placement of the sensors is left upon the discretion of the embedded developer since the critical boundary can vary from device to device. The sensors are placed randomly on the chip such as to receive more general statuses and not to localise the sensors to a particular area on the chip. The status values of the 16 DSs in nominal, clock-glitch fault injection and electro-magnetic fault injection case is presented in Figure 3. Please note that each sensor has a buffer depth of 13 which is represented in the x-axis for all the 16 DSs.

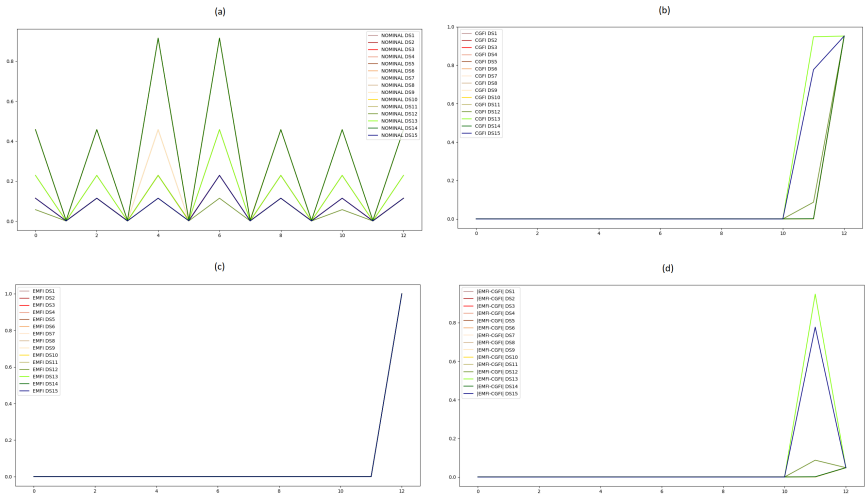

Fig. 3: Comparison between states of 16 DSs for nominal as well CGFI and EMFI cases. The x-axis is the status buffer for each DS. (a) represents the nominal state of the DSs when no injection is performed, (b) represents the state of the DSs when CGFI is performed, (c) represents the state of the DSs when EMFI is performed, and (d) represents the difference in values of the DSs from CGFI and EMFI cases. It can be seen that some DSs behave similarly in both EMFI and CGFI cases (for example DS12). In case of one DS based system, it would not have been possible to differentiate between the type of attack. Therefore, sensor aggregartion provides more features which can be utilized by a classification algorithm to differentiate between the type of attack.

The classification task is divided into four parts, as mentioned below, and each of them is inferred separately over the same ML models.

i. Detection of EMFI by performing binary classification between nominal and injected classes of EMFI dataset

ii. Detection of CGFI by performing binary classification between nominal and injected classes of CGFI dataset
iii. Combining the FI datasets of both EMFI and CGFI and classification between the combined FI datasets and nominal dataset from both EMFI and CGFI sessions
iv. Classification between EMFI and CGFI datasets to detect the perturbation type (by combining both FI datasets and classifying between them)

3.5 Hardware testing of the design using HLS

The performance of the design is validated on hardware. To achieve this, the authors utilize the HLS methodology for a quick design evaluation with real digital sensor values on a FPGA. The motivation is to replicate the software results on the target hardware platform, which is the actual working environment for the design, by benchmarking with the offline performance. For HLS we use the Xilinx Vivado HLS 2019.2 tool where the target FPGA is a Digilent Arty S7-50 board, which is compatible with the Vivado HLS design suite.

Methodology The framework is composed of four main stages and can be understood from the list below:

i Firstly the training is carried out to train the ML model parameters using standard software ML frameworks in any high level language (in our case Python) after which the learnt parameters are extracted.
ii A C/C++ implementation of the design is created from scratch with no external library support and the ML inference model is initialized with the learnt parameters extracted at stage **i**.
iii The C source, along with a testbench written in C to validate the design performance at simulation level, is used to perform HLS using Vivado HLS design platform to generate RTL (Register Transfer Level) without any additional optimization, such as the usage of HLS pragmas, other than the ones already integrated in the design flow, thereby having no control upon the generated HDL (Hardware Description Language) code structure.
iv The generated RTL is packed into an IP and imported in the Vivado HLX suite where a controller program is written in Verilog to interface with the IP. Real DS test dataset for the ML IP is stored in a RAM block which is used by the controller program to segment and test the IP for classification accuracy. (please see figure 4.)

Experimental setup The setup is simply composed of a computer connected to the target FPGA board. The target board, Digilent Arty S7-50, is a Vivado compatible FPGA board. The design is run at a clock-cycle of 10 nanoseconds (ns) with on-board clock running at 100 MHz frequency. Other features of the FPGA include 52,160 logic cells, 8,150 slices, 65,200 Flip-Flops, 2,700 Kbits of Block RAM, and 120 DSP slices.

Hardware Performance The test dataset (test vectors) used to validate the classification accuracy of the ML IP created using HLS is same as used at the software level in the ML testing phase. Upon evaluation, the classification capability remains unaltered at the hardware level. The resource utilization report for the current HLS implementation is shown in Table 1.

Resource	Utilization	Available	Utilization (%)
LUT	8898	32600	27.29
LUTRAM	266	9600	2.77
FF	8478	65200	13.00
BRAM	1.50	75	2.00
DSP	43	120	35.83
IO	6	210	2.86
BUFG	1	32	3.13

Table 1: Post-implementation resource utilization of the FPGA for the whole design including the controller module and test data

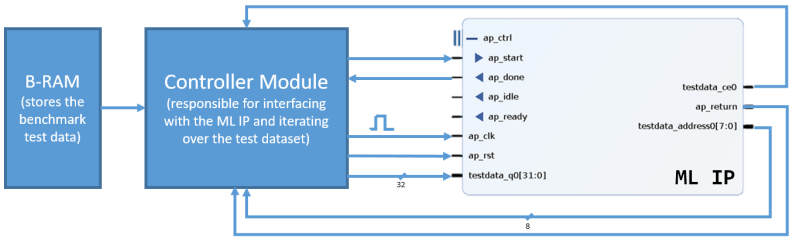

Fig. 4: High level block diagram of the test setup with a controller module interfacing the ML HLS IP with the benchmark test data stored in a B-RAM

Each DS consumes ≋ 400 LUT slices. In this design we chose the minimum number of DSs required to achieve sensor aggregation and improved accuracy fo the classification. We use 16 DSs which is approximately 6400 LUT slices. The SM consumes 8898 LUT slices which is 39% greater than the total consumption of all the deployed DSs. However, it is to note that the SM design is not optimized and an optimized design can be similar in area of the total number of DSs deployed. In terms of throughput, the total number of cycles required to perform sensor aggregation and one classification, with a DS buffer length of 13 statuses, is 38275 cycles. It is the first time an online integrable sensor aggregation strategy and ML based classification for fault injection detection is showcased.

The FPGA test is performed to justify the use-case and establish a proof of concept. However, the next task would be to optimize the HLS output such

that the footprint on the hardware fabric is further reduced and throughput is enhanced.

4 Results

In this section we present all the results of the four classification categories at the software level. Table 2 shows the accuracy percentages of the various classifications performed. Since all the classifications are binary, two datasets are used for all cases. The attack diversity column of the table refers to the different conditions in which the data were recorded viz. four chip locations for EMFI and two different conditions for CGFI. If both the datasets are used for either classification from nominal condition or between both the attack conditions, the diversity is indicated as 6. For classification between the types of perturbation the DS saturation leads to class overlapping and, thus, the ML model tends to predict incorrect classes.

Classification between		**Attack**	**Detection Accuracy (%)**		
Data A	**Data B**	**Diversity**	**True detection**	**False Pos.**	**False Neg.**
EMFI	Nominal	4	98.51	0	1.49
CGFI	Nominal	2	100.00	0	0
∗EMFI+CGFI	Nominal	6	91.98	0	8.02
∗∗EMFI	CGFI	6	77.25	22.74†	0

†This value denotes the percentage of tests where the ML model predicts CGFI for EMFI cases. This is due to the DS saturation phenomenon explained in Section 3.4. ∗Stage 1 detection result, ∗∗Stage 2 detection result

Table 2: Detection accuracy of the best performing ML model in various classification tasks

The evaluation is extended to compare the ML based methods with the classical sensor threshold based method where a sensor is tuned to set a threshold value that defines the class boundary. The disadvantage of this method is in choosing the threshold value, which is often chosen empirically over multiple test cases, therefore leading to non-generic coarse-tuned setting which may fail due to lack of sufficient test cases. To overcome this, we train the threshold for each individual DS and use the collective result to predict the class i.e. if ≥ 8 (half of total number of sensors involved) DSs predict positive, the result is taken as FI, else nominal. The result comparison and gain of ML method over classical threshold based method is provided in the following subsections.

4.1 Threshold optimization of every DS

To optimize the thresholds for each DS, the buffer data (13×4 bytes) is converted to an average form as shown below in equation 1, where X is the input vector

(of size 13) and X' is the vector obtained after the averaging process:

$$\forall i \in \{0, 1, ..., 12\} \text{ and,}$$
$$X = \{V_i\} \text{ and } X' = \{V_i'\}, \text{ where,}$$
$$V_i' = V_i \text{ if } i = 0, \text{ else } V_i' = (V_i + V_{i-1})/2. \quad (1)$$

Thereafter the bounds (Lower/Upper for class Zero/One) of both the classes (0: Non-injection/Nominal, 1:Injection) are calculated by running a linear search over 80% (similar to training set ratio in ML methods) of the dataset. These bounds are used as threshold for the test set to detect FI states.

4.2 Classification between EMFI and Nominal condition

The EMFI is performed with a very low power. The aggregated sensor method enables high precision detection of FI. The detection performance of all the ML models is shown in Table 3 along with a comparison with the classical threshold based method. The accuracy fluctuation between the different datasets corresponds to the DSs' behavior based on the locality of EMFI.

4.3 Classification between CGFI and Nominal condition

The best performing ML model (GNBC: see Table 3) is chosen for further evaluation to maintain symmetry along all the evaluations. Similar comparison of performance between the ML and Threshold method, shown in Table 4, is made for the CGFI datasets. The ML method produces 100% accurate predictions over the test dataset since the FI case produces nearly full saturation in the DSs' statuses.

	ML Models				
Dataset	**MLP**	**SVM**	**LRC**	**GNBC**	**Threshold**
EM1	94.75	92.25	81.50	97.50	60.92
EM2	95.71	92.93	88.64	97.98	81.17
EM3	98.90	98.90	90.63	99.72	91.23
EM4	100.00	100.00	100.00	100.00	100.00
Combined(EM1-EM4)	96.25	96.18	88.49	**98.51**	82.82

Table 3: Detection accuracy of different ML models over the EMFI datasets. (EM1–EM4 correspond to the different recording sessions with varying injection location on the XY plane of the DUT). (Note: All values in the table are in %)

4.4 Classification between combined EMFI and CGFI against Nominal condition

In this case the attack datasets from both EMFI and CGFI sessions are combined to form the attack class. Similarly, the nominal class is also created for training/inference of the binary classifiers. The evaluation is also performed with the

	Detection Accuracy (%)	
Dataset	**ML (GNBC)**	**Threshold**
CG1	100	98.64
CG2	100	99.27
CG1&CG2 combined	100	98.24

Table 4: Detection accuracy comparison of CGFI between ML and Threshold methods over CGFI datasets

threshold method and the results are compared as shown in Table 5. The combination of diverse datasets increases the linear classification complexity manifold which can be observed from the performances of the ML and Threshold models over the hybrid datasets.

	Classification Accuracy (%) of nominal dataset from		
Method	**EMFI**	**CGFI**	**EMFI+CGFI**
ML (GNBC)	98.51	100	91.98
Threshold	82.82	98.24	89.36

Table 5: Detection accuracy of FI from nominal case with combined EMFI and CGFI attack case. (The EMFI and CGFI columns contain results of combined dataset of all sessions)

4.5 Classification based on attack type between EMFI and CGFI

Finally, for perturbation detection, the results are presented in Table 6. It is important to note that this classification is performed with just the attack case (injection datasets only) and the DSs, as mentioned earlier, have linear saturation directly proportional to attack strength. Thus, in the attack dataset with the two classes being CGFI and EMFI, the difference is minimal where the linear classifiers tend to fail in precisely optimizing the separation plane.

	Values in %		
Method	**Accuracy**	**False Pos.**	**False Neg.**
ML (GNBC)	77.25	22.75	0
Threshold	39.61	0	60.39

Table 6: Accuracy comparison of perturbation detection between EMFI and CGFI of ML and Threshold methods

5 Conclusion

In this work we introduce a two-stage fault injection detection framework for EMFI and CGFI while opening doors to more attack sources. Furthermore, we provide analysis of fault detection for individual attacks of EM and CG while comparing the ML based model with a classic threshold based method where the threshold is trained instead of manually choosing a value. The ML model is evaluated on a FPGA with benchmark testing, using the same test dataset from the software evaluation, to record the classification accuracy of the ML model on hardware. The ML model performs very well in detecting individual attacks with accuracies of 98.51% and 100.00% for EMFI and CGFI respectively, and 92% in detecting both CGFI and EMFI combined. Furthermore, to detect the perturbation type, classification is performed between CGFI and EMFI fault dataset only with 77.25% detection accuracy.We finally combine all the methods to form a two-stage FI detector where the stage one predicts the existence of an attack when the DSs input can either be nominal, EMFI or CGFI. The second stage is enabled if the first stage detects an attack and, thus, it predicts the perturbation type.

References

1. Bar-El, H., Choukri, H., Naccache, D., Tunstall, M., Whelan, C.: The sorcerer's apprentice guide to fault attacks. Proceedings of the IEEE **94**(2), 370–382 (2006)
2. Barthe, G., Dupressoir, F., Fouque, P.A., Grégoire, B., Zapalowicz, J.C.: Synthesis of fault attacks on cryptographic implementations. In: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security. pp. 1016–1027. ACM (2014)
3. Biham, E., Shamir, A.: Differential fault analysis of secret key cryptosystems. In: Annual international cryptology conference. pp. 513–525. Springer (1997)
4. Boneh, D., DeMillo, R.A., Lipton, R.J.: On the importance of checking cryptographic protocols for faults. In: International conference on the theory and applications of cryptographic techniques. pp. 37–51. Springer (1997)
5. Breier, J., Bhasin, S., He, W.: An electromagnetic fault injection sensor using hogge phase-detector. In: 2017 18th International Symposium on Quality Electronic Design (ISQED). pp. 307–312 (March 2017)
6. Breier, J., Hou, X., Bhasin, S.: Automated Methods in Cryptographic Fault Analysis. Springer (2019)
7. Cayrel, P.L., Colombier, B., Drăgoi, V.F., Menu, A., Bossuet, L.: Message-recovery laser fault injection attack on the classic mceliece cryptosystem. In: Canteaut, A., Standaert, F.X. (eds.) Advances in Cryptology – EUROCRYPT 2021. pp. 438–467. Springer International Publishing, Cham (2021)
8. Claudepierre, L., Péneau, P.Y., Hardy, D., Rohou, E.: TRAITOR: A Low-Cost Evaluation Platform for Multifault Injection. In: ASIACCS 2021 - 16th ACM ASIA Conference on Computer and Communications Security. pp. 1–6. ACM, Virtual Event Hong Kong, Hong Kong SAR China (Jun 2021)
9. Dottax, E., Giraud, C., Rivain, M., Sierra, Y.: On second-order fault analysis resistance for crt-rsa implementations. In: IFIP International Workshop on Information Security Theory and Practices. pp. 68–83. Springer (2009)

10. Hospodar, G., Gierlichs, B., De Mulder, E., Verbauwhede, I., Vandewalle, J.: Machine learning in side-channel analysis: a first study. Journal of Cryptographic Engineering **1**(4), 293 (2011)
11. Joye, M., Tunstall, M.: Fault analysis in cryptography, vol. 147. Springer (2012)
12. Menu, A., Bhasin, S., Dutertre, J.M., Rigaud, J.B., Danger, J.L.: Precise spatio-temporal electromagnetic fault injections on data transfers. In: 2019 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC). pp. 1–8. IEEE (2019)
13. Miura, N., Najm, Z., He, W., Bhasin, S., Ngo, X.T., Nagata, M., Danger, J.: Pll to the rescue: A novel em fault countermeasure. In: 2016 53nd ACM/EDAC/IEEE Design Automation Conference (DAC). pp. 1–6 (June 2016)
14. Moore, S.W., Anderson, R.J., Kuhn, M.G.: Improving smartcard security using self-timed circuit technology. In: Fourth ACiD-WG Workshop, Grenoble (2000)
15. Murdock, K., Oswald, D., Garcia, F.D., Van Bulck, J., Gruss, D., Piessens, F.: Plundervolt: Software-based fault injection attacks against intel sgx. In: Proceedings of the 41st IEEE Symposium on Security and Privacy (S&P'20) (2020)
16. Proy, J., Heydemann, K., Majéric, F., Cohen, A., Berzati, A.: Studying em pulse effects on superscalar microarchitectures at isa level. arXiv preprint arXiv:1903.02623 (2019)
17. Qiu, P., Wang, D., Lyu, Y., Qu, G.: Voltjockey: Breaching trustzone by software-controlled voltage manipulation over multi-core frequencies. In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security. pp. 195–209 (2019)
18. Riviere, L., Najm, Z., Rauzy, P., Danger, J.L., Bringer, J., Sauvage, L.: High precision fault injections on the instruction cache of armv7-m architectures. In: 2015 IEEE International Symposium on Hardware Oriented Security and Trust (HOST). pp. 62–67. IEEE (2015)
19. Saha, S., Alam, M., Bag, A., Mukhopadhyay, D., Dasgupta, P.: Leakage assessment in fault attacks: A deep learning perspective. Cryptology ePrint Archive, Report 2020/306 (2020), https://ia.cr/2020/306
20. Saha, S., Jap, D., Patranabis, S., Mukhopadhyay, D., Bhasin, S., Dasgupta, P.: Automatic characterization of exploitable faults: a machine learning approach. IEEE Transactions on Information Forensics and Security **14**(4), 954–968 (2018)
21. Selmane, N., Bhasin, S., Guilley, S., Danger, J.L.: Security evaluation of application-specific integrated circuits and field programmable gate arrays against setup time violation attacks. IET Information Security **5**(4), 181–190 (December 2011)
22. Swaminathan, S., Chmielewski, L., Perin, G., Picek, S.: Deep learning-based side-channel analysis against aes inner rounds. IACR Cryptol. ePrint Arch **2021**, 981 (2021)
23. Tang, A., Sethumadhavan, S., Stolfo, S.: {CLKSCREW}: exposing the perils of security-oblivious energy management. In: 26th {USENIX} Security Symposium ({USENIX} Security 17). pp. 1057–1074 (2017)
24. Taouil, M., Aljuffri, A., Hamdioui, S.: Power side channel attacks: Where are we standing? In: 2021 16th International Conference on Design & Technology of Integrated Systems in Nanoscale Era (DTIS). pp. 1–6. IEEE (2021)
25. Zaid, G., Bossuet, L., Dassance, F., Habrard, A., Venelli, A.: Ranking loss: Maximizing the success rate in deep learning side-channel analysis. IACR Transactions on Cryptographic Hardware and Embedded Systems pp. 25–55 (2021)
26. Zankl, A., Seuschek, H., Irazoqui, G., Gulmezoglu, B.: Side-channel attacks in the internet of things. In: Research Anthology on Artificial Intelligence Applications in Security, pp. 2058–2090. IGI Global (2021)