Inducing Local Timing Fault Through EM Injection

Marjan Ghodrati, Bilgiday Yuce, Surabhi Gujar, Chinmay Deshpande,
Leyla Nazhandali, Patrick Schaumont
Virginia Tech, Blacksburg, VA, USA

ABSTRACT

Electromagnetic fault injection (EMFI) is an efficient class of physical attacks that can compromise the immunity of secure cryptographic algorithms. Despite successful EMFI attacks, the effects of electromagnetic injection (EM) on a processor are not well understood. This paper presents a bottom-up analysis of EMFI effects on a RISC microprocessor. We study these effects at three levels: at the wire-level, at the chip-network level, and at the gate-level considering parameters such as EM-injection location and timing. We conclude that EMFI induces local timing errors implying current timing attack detection and prevention techniques can be adapted to overcome EMFI.

CCS CONCEPTS

• **Security and privacy** → **Hardware attacks and countermeasures**;

KEYWORDS

hardware security, fault attack, electromagnetic radiation

ACM Reference Format:

Marjan Ghodrati, Bilgiday Yuce, Surabhi Gujar, Chinmay Deshpande, Leyla Nazhandali, Patrick Schaumont. 2018. Inducing Local Timing Fault Through EM Injection. In *DAC '18: DAC '18: The 55th Annual Design Automation Conference 2018, June 24–29, 2018, San Francisco, CA, USA*. ACM, New York, NY, USA, 6 pages. <https://doi.org/10.1145/3195970.3196064>

1 INTRODUCTION

The extent of embedded systems role in modern life has continuously increased over the years. Moreover, embedded systems are assuming highly critical functions with security requirements more than ever before. Physical attacks can compromise the effectiveness of the most secure cryptographic algorithms by manipulating or just observing the hardware that implements them. There are two types of physical attacks, side-channel attacks, and fault attacks. Unlike side-channel attacks which are passive, fault attacks are active. In these attacks, the attacker manipulates the working conditions of the circuit to the point that it deviates from its correct functionality [6]. This can result in either bypassing the security checks or creating new side-channels that help uncover the cryptographic keys. There are several techniques for fault injection such as clock

glitching [5], voltage starvation [21], optical injection [19], temperature changes [17], micro probing [18] and electromagnetic fault injection [8] [12]. The most common forms of fault attacks are clock and voltage glitching. However, designers have proposed several countermeasures against these specific types of fault attacks which have rendered them unusable. For another popular form of fault attack, optical injection, the attacker needs to decapsulate the chip, which is a complicated process with the risk of damaging the target device. Moreover, it is possible to employ countermeasures such as light sensors to detect such attacks. There has been an increasing interest in EMFI in the past several years as there is no need to decapsulate the chip. Moreover, with EMFI, the attackers have a broad range of options such as radiation intensity as well as offset and duration of the radiation at their disposal. These options allow the adversaries to fine tune their attack to quickly achieve their desired faulty behavior. Despite its growing popularity, limitations and capabilities of this attack are not very well understood. Several detection techniques have been proposed, but most of them are either very expensive to implement or not very effective. We believe to properly model EMFI and develop effective countermeasures, a deeper understanding of the EM effect on a chip is needed. In this paper, we try to perform a bottom-up analysis of EM fault injection on a RISC microprocessor and do a comprehensive study at all wire-level, chip-network level and gate-level. The contributions of this paper are as follows:

- We provide experimental results on the effect of EM injection on different wires and networks on the chip. We particularly study the effect on three main networks on the chip, namely clock, power and reset. We show that the clock is the most affected network among the three.
- We study how the wire and chip level effects translate into gate-level ones. Using several experiments and data analysis, we conclude that most of the faults happen when the EM has induced a glitch in the clock network.
- Finally, we provide a study that shows the level of effectiveness and limitations of a typical clock glitch monitor to detect EM injection. Based on this analysis we propose techniques for enhancing the detection rate.

The rest of the paper is organized as follows. Section 2 provides the experimental setup used for implementing EMFI and introduces the target device. Section 3 presents the effect of EM on single wires or pins, while Section 4 discusses the same thing for electrical networks of a microprocessor, namely power, clock and reset. Section 5 describes the gate-level effects of EM. We provide a countermeasure method in Section 6. Finally, we present related work in Section 7, and conclude in Section 8.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

DAC '18, June 24–29, 2018, San Francisco, CA, USA

© 2018 Association for Computing Machinery.

ACM ISBN 978-1-4503-5700-5/18/06...\$15.00

<https://doi.org/10.1145/3195970.3196064>

Figure 1: Block diagram of Fault Injection Setup. (1) Inspector (software running on PC) transmits EMFI parameters to the Glitch Controller and arms it. (2) Inspector sends the command to the board to start the operation (3) Microprocessor sends a *trigger* to the Glitch Controller. (4) Glitch Controller makes EMFI-probe inject EM pulse after a predetermined delay (offset) (5) The output is sent back to Inspector.

2 EXPERIMENTAL SETUP

In this section, we introduce our target device, EM injection setup, and influential parameters affecting EMFI.

2.1 Target Device

The device under attack is a chip with an in-order RISC processor implementing the SPARC-V8 instruction set. The processor is manufactured in 180 nm TSMC process on a $5 \times 5 \text{ mm}^2$ packaged die. The processor core is based on the open-source 32-bit LEON3-design[1]. In addition to the RISC core, the prototype chip includes 2KB data cache and 1KB instruction cache. It also provides a 4-pin General Purpose Input Output (GPIO) port with interrupt capability and a Universal Asynchronous Receiver/Transmitter (UART) interface. The chip mounts on a 108-pin Pin Grid Array (PGA) package, which is socketed on a custom testing board which is developed as an extension to the SAKURA-G Board [4] and attaches to it using the standard expansion headers. There are two reasons why we use a custom-made processor as our target device as opposed to an off-the-shelf microprocessor:

Knowledge of layout: We have the detailed information about the layout including the location of the clock network, memories, various pipeline flip-flops (FFs), etc. This helps us explain why attacking the processor at various points on the chip results in different outcomes.

Extra observability: We have designed our processor such that we can observe its state in its most detailed form. All the pipeline FFs on the processor are memory-addressable and therefore, readable using the program. Furthermore, we have the option to copy the state of all the FFs in an identical set of shadow FFs utilizing an input signal that we control from outside the chip. This allows us to take a snapshot of the processor’s state at any desired time, particularly right after the EM injection.

2.2 Fault Injection Setup

Figure 1 gives an overview of the setup. It consists of a PC, target device described in the preceding subsection and Riscure Fault Injection equipment. The PC executes two software programs in parallel to manage the entire injection process: Inspector [2] software which is developed by Riscure to control the EM injection

Figure 2: Picture of our experimental setup.

equipment and GRMON software that communicates with the target microprocessor.

We control the timing of the injection using a sophisticated process between the target microprocessor, PC, and the glitch-controller [3], which is part of the Injection equipment. First, we decide during which instruction of the program we would like to inject the fault. Just before that cycle, we add an instruction to write a '1' to the GPIO pin that is connected to the glitch-controller which is used as its trigger. Upon receiving this trigger, the glitch-controller waits for a predetermined amount of time (offset) after the rising edge of the clock and then triggers the probe to create a voltage pulse that sends current through the coil to produce a magnetic field. The intensity of the EM injection depends on the magnitude of the voltage pulse. The XYZ location, intensity and the offset of the EM injection are all predetermined by the attacker using the Inspector software. Figure 2 shows the picture of our experimental setup.

2.3 EM injection parameters

In this section, we present influential parameters affecting EM injection and describe how we determined the suitable range of values for them:

Injection location: Through extensive sweeping of the microprocessor, we learned that some of the points on the processor are so sensitive that push the processor to an unresponsive state. Although this forms a denial of service attack, it is not desirable in a fault attack. We excluded those points from our experiments.

Injection intensity: We represent intensity as a percentage of the highest injection voltage, which is 450V. For example, an EMFI with 45V pulse is considered at 10% intensity. Not surprisingly, very low-intensity injections do not affect the target, and in some cases, very high-intensity injections take the processor to an unresponsive state. We excluded these very low and very high intensities from most of our experiments.

Injection probe type and distance from chip: The Riscure equipment comes with four probes with different diameters. We use a probe with a 1.5mm diameter tip. The probe is aligned on top the die surface with a vertical distance of approximately 2 mm. We chose these parameters as they can create faults without causing an unresponsive state.

Glitch offset: Glitch offset refers to the timing of EMFI relative to the rising edge of the clock and takes values between 0 – 41.6ns

Figure 3: Effect of EM glitch on a pin. (a) powered-down circuit (b) powered-up circuit

where 41.6 ns is the clock period of the microprocessor. We will later see that the glitch offset has a significant effect on the attack success.

3 WIRE-LEVEL EFFECTS OF EM INJECTION

An EM injection affects the victim circuit in a fashion that is very similar to a transformer's operation. The coil on the EM probe acts as the primary winding of the transformer. The wires inside the chip form some loops that serve as the secondary winding of the transformer. During EM injection, a variable current flows through the coil, due to coupling some current is generated on the wires. This, in turn, results in changes in the voltage of the wires.

Figure 3a shows the effect of the EM injection on one of the pins of a chip that is not powered up. The blue waveform shows the coil current, and the red waveform is the perturbation on the mentioned pin. When the current in the coil ramps up, the wire's voltage is positive while when the current goes down, its voltage is negative. Based on Faraday's law, the magnitude of this voltage depends on the rate that the probe's current fluctuates. As the probe's current dampens, so does the voltage on the wire.

Figure 3b shows the same effect on one of the pins when the chip is powered up. The observed pin is a General purpose I/O (GPIO) pin that the processor is driving it to logic '0'. When the current increases, the pin's voltage shoots higher to the point that it can be considered logic '1'. As the current in the coil decreases, the voltage drops back to the ground.

Figures 4a-d show the same effect on a clock net. The injection is happening in the low phase of the clock. The difference between the figures is the intensity of injection. While at 30% intensity (Figures 4a), the difference is merely a nudge, at 70% (Figures 4d) a full-grown glitch is observable.

In a digital circuit, the previously observed phenomenon translates into a more specific effect: When the net carries logic 1 (VDD), increasing the current in the coil, does not have any impact as the voltage is capped at VDD, but decreasing of the current can lower the net's voltage. If the electromagnetic field is powerful enough, it can even push the net to logic 0 (GND). On the other hand, when the net carries logic 0 (GND), decreasing the coil current does not affect the voltage, but increasing it can increase the net's voltage, sometimes turning the net to logic 1 (VDD). Figures 5a-b captures this phenomenon. In (a), the attack is done while the clock is high and in (b), the attack is performed when the clock is low. In (a), there is no noticeable change in the voltage while the coil's current

Figure 4: Effect of glitch intensity on the EM perturbation. (a)30%, (b)40%, (c)60%, (d)70%

Figure 5: Effect of EM injection on the clock. Two plots show injection during (a) high phase and (b) low phase of clock.

increases, but the voltage drops to GND as the current decreases. In (b), the clock net is pushed to VDD as the current increases and drops back to GND when the current decreases.

4 CHIP-LEVEL EFFECTS OF EM INJECTION

In the previous section, we showed the impact of injection on individual wires/pins on the chip. During an EM injection, the electromagnetic field is the strongest right under the probe. As we move further from the probe, the EM is reduced and so does its effect on the wire regarding altering the voltage. Nevertheless, EM can use the global networks on the chip to extend its influence. In this section, we study such effects.

There are three global networks on a typical chip: power network, clock network, and reset network. The power network is designed such that it withstands noise and other coupling effects. It is excessively redundant to ensure there is always enough power to keep the circuit in working condition without slowing down. Designers perform elaborate voltage drop analysis to find weak points in the network and correct it by adding more power pins, power straps or power rings. With such sophisticated design, it is not surprising that when we examine power or ground pins that are relatively far from the point of attack, we cannot observe any effect.

Figure 6: Effect of the EM injection on top of the reset pin

In other words, the power network does not allow the influence of a single affected power pin to propagate to the rest of the chip.

Another global network is the clock network. As for the clock, it is designed to ensure a high speed in distributing the clock. There are numerous buffers on the clock tree to boost the signal and allow fast transfer of the clock voltage to all corners of the chip. We expect that this works in favor of an attacker. If an attacker affects the root of the clock tree or one of the nodes close to the root, we expect that the malicious changes in the voltage will propagate quickly throughout the chip. Unlike power network where we have access to many power and ground pins around the chip, we are not able to directly observe the clock in various locations to verify our hypothesis. Instead, we will use circumstantial clues to support this theory, which we will present in the next section.

Finally, the reset network is used to reset all the FFs in the design to zero. A typical reset network is not as fast as a clock network to reduce area and power consumption. On our chip, the reset is active low and in its inactive state is driven by a high current voltage regulator which does not allow the reset pin to deviate from high voltage. Figure 6 shows the effect of the EM injection on top of the reset pin. We can see that the reset voltage does not change to the point that causes a reset. It is possible that inside the chip, local reset wires get affected by EM injection and induce a "reset" on some FFs. We will consider this possibility in the next section.

5 GATE-LEVEL EFFECTS OF EM INJECTION

In this section, we make an effort to explain how the EM effects at the wire/chip level manifest themselves at the gate-level to the point that puts the microprocessor in a faulty state. Based on the observations in the previous two sections, we can envision three scenarios that turn the processor's state into an erroneous state:

1: Some of the inputs of the FFs swing to the incorrect logic, and while they are in this state, the positive edge of the clock arrives, and the FF captures this wrong value.

2: The EM injection activates the reset wire for some of the FFs, and that forces those FFs to go to logic '0', which can be a wrong value for some of those FFs. Since our design uses a synchronous reset, this can also only happen at the rising edge of the clock.

3: The EM injection creates a clock glitch. Similar to a timing attack, the FF opens its gates and latches the values before the correct output has been calculated resulting in a faulty state.

We think that one or more of the above situations might be at work when a fault happens. However, we do not have access to internal wirings of the chip and cannot prove or disprove any of the above cases by observing them. Nevertheless, we can use some indirect clues to conclude which ones are more likely to be the source of the errors. Based on our experiments, we believe most

of the EM injection faults are caused due to glitches in the clock network. We provide two arguments to support our claim:

5.1 Temporal argument: Effect of the timing of the injection

Figure 7 shows the clock under attack with varying offset values. The numbers on the top left corner of each plot shows the ratio of the number of corrupted FFs to the total number of FFs. These results are the average for several points on the chip. We can see that the effect varies from no glitches (cases (b), (c) and (d) with 0% faulty FFs), to one glitch (cases (a) and (e) with 44% and 23% faulty FFs), and finally to two glitches (case (f) with 72% faulty FFs). All of these cases can be explained by the phenomenon described in Section 3. If any of scenarios 1 or 2 described earlier in this section were true, we should only observe faults for case (f), where the attack happens shortly before the rising edge of the clock. Nevertheless, we can see that the number of faults is non-zero for cases (a) and (e) despite the fact that the injection is not close to the rising edge. Instead, the number of faults becomes 0 when the clock is not affected, namely cases (b), (c), and (d). These results strongly suggest that the EM faults happen due to disturbances in the clock caused by EM.

5.2 Spatial argument: Effect of the attack location relative to the clock tree

Figure 8 presents an image of our chip layout where the top levels of the clock and all the FFs are highlighted. It also shows two points where we inject EM to explore the effects of EMFI on different locations on the clock tree. While the injection point A is very close to the root of the clock, point B sits at much lower branches of the clock tree (level 15). As discussed in Section 2, we can take a snapshot of the processor state using a trigger, which is set one clock cycle after EM injection. In our program, we read the content of these FFs and record them in a file. We create two files, one after the attack and one for correct operation. Using some scripts, we discover all the faulty FFs. Figure 9a-b shows the corrupted FFs on the layout for each of the attack points described earlier. When we inject the EM close to the root of the tree, there are 146 affected FFs, and they are spread all over the chip whereas, for the other case, there are only 24 corrupted FFs, and they are compacted around the injection point. Based on these results, we conclude that the clock tree is propagating the glitch effect from the higher levels to the lower branches of the clock tree, confirming that the EM attack, in essence, is a timing attack. We verified this observation at several other points on the chip with different programs, and we always observe similar outcomes.

6 EM COUNTERMEASURE

Based on our experimental observations and analysis, we could postulate that EMFI induces local timing faults in a chip. To further validate our hypothesis, we integrated a configurable timing sensor into our microprocessor. This timing sensor monitors the propagation delay in the critical path of the circuit and raises the alarm if there are violations of the setup time constraints [9].

We started our experiments by performing EMFI on 20 different locations on the processor, where each location was attacked 1000 times. We found that the timing sensor was able to detect 100% of

Figure 7: Glitch offset effect on clock. (a) offset = 0, (b) offset = 10, (c) offset = 20, (d) offset = 25, (e) offset = 30, (f) offset = 40

Figure 8: Clock tree, observable FFs and two injection points on the layout of the processor

Figure 9: Faulty FFs on layout (a) injection point on root of clock tree, (b) injection point on lower levels of clock tree

the injected faults on some locations whereas it could not detect any faults on few other locations. We noticed that the sensor could detect the faults when EM was injected around the root of the clock tree as shown in Figure 9a. On the contrary, it could not detect any faults when the injection point is on the lower levels of the clock tree similar to the case shown in Figure 9b.

With this observations, we could make the inference that when the fault is injected at the top level of the clock tree, the glitch effect gets propagated to the lower levels including the branch that timing sensor resides. As a result, the timing sensor can detect it. However, when the fault is injected at the lower levels of the clock tree, the effect does not propagate and therefore remains local. This means the timing sensor observes a healthy clock and assumes no fault has happened.

Suitable timing sensor for EMFI: Based on the above observations and analysis, we can interpret that to increase the percentage of EM fault detection, the number of timing sensors in the chip should increase. Their placing is also very important. The designer needs to make sure that no matter how far from the root of the clock the injection happens, there is a timing sensor in the vicinity that can catch the fault. The number of needed timing sensors and their locations is part of our future work.

Detecting faulty state versus detecting EM injection: Even at a proper offset, location, and intensity, an EM injection may not be successful to induce faults due to noise effects. For example, injecting EM at point A (from Figure 8), with 80% intensity and 40 ns offset, out of 1000 injections, only 720 of them push the processor into an incorrect state. Our timing detector can catch all these 720 cases and does not raise the alarm for the remaining 280 injections. On the contrary, if we had used a coil to detect the presence of EM field, the alarm would have been raised for all those 1000 times regardless of whether the injection is causing a fault or not. Depending on the application, one or the other might be desirable. For a typical design where both security and speed are essential, the timing detector is preferable as it prevents wasted cycles during those cases that no fault has occurred. Nevertheless, for some devices where security has the utmost importance in design, the designer might prefer to catch an attack even if it is not causing a fault yet.

7 RELATED WORK

J.-J. Quisquater and Samyde [16] presented the first instances of EMFI in 2002. Later in 2007, [12] used high frequency spark gaps instead of magnetic field to induce current which leads to a very strong electromagnetic burst and radiation. They used this method to extract the key of a CRT-based RSA algorithm.

Dehbaoui et. al. in [8] used EM pulses to implement a successful attack on a software implementation of AES on an 8-bit microcontroller and a hardware AES embedded in a FPGA. Based on their observation, they draw a hypothesis that EM pulses induce timing violations. Nevertheless, they do not back their hypothesis with cludes. They also added a timing sensor on the FPGA that implemented AES. During their experiments, they noted the sensor was only able to detect 10% of induced faults. The paper lacks analysis of why timing sensor is successful in some cases and not for others.

A year later, Moro et. al. in [14] explored the impact of an electromagnetic glitch fault injection on a state-of-the-art microcontroller and built an associated register-transfer level fault model. According to this paper, EM pulses can affect both data flow and program flow. They showed some instructions or registers are more vulnerable than the others, but due to lack of access to the architecture of the microprocessor, their fault model is not comprehensive.

The authors in [15] created an experimental setup for performing fault injection and were able to provide the preliminary nature of the induced fault model. They experimentally demonstrated that the faults were very local and produced a combination of bit-set/reset faults.

Perhaps the closest related work to our research is the study conducted by Zussa et. al [20]. They proposed the use of delay-based glitch detectors to detect EMFI. This method identifies setup time violations by using fixed guarding delay which is set slightly higher than critical path of the design. Hence, voltage disturbances will be detected, and an alarm will be issued. They implemented five glitch detector on a FPGA, and they reported detection ratio of 32%. The glitch detector is inadequate with a best-case detection rate of about 32%. Despite evaluating the effectiveness of timing sensor to detect EMFI, no study or explanation is provided on why the detection rate is low, and how to improve it.

Miura et al. introduced PLL-based sensor circuit to detect EMFI reactively [13]. [10], [7], [11] all present some form of countermeasure. Some of these papers assume EMFI always affects the clock globally, which we showed is not true. Furthermore, none of them provides a detailed analysis of how the EMFI works and why or why not their detector is effective.

8 CONCLUSION

Despite EMFI's increasing popularity, their effects are still not well understood, and our work is an effort to unravel the way an EM attack affects a microprocessor on different levels of abstraction. This knowledge is necessary to develop accurate fault models and design effective countermeasures. From our experiments, we saw that these effects were most apparent in the clock network when compared to the other two global networks, namely, power and reset. We observed that EMFI causes clock glitches. Furthermore, we showed that relative delay and position of the EM injection with respect to the clock plays a significant role in the effectiveness of

EMFI. In conclusion, our work suggested that EM attacks behaved similar to timing attacks and could propagate using the clock tree network. Thus, with some modifications, we believe that the current countermeasures for timing attacks like timing detectors can be used to thwart EM attacks.

9 ACKNOWLEDGEMENT

This research was supported in part by National Science Foundation Award 1441710.

REFERENCES

- [1] [n. d.]. LEON3 processor. <http://www.gaisler.com/index.php/products/processors/leon3>. ([n. d.]). [Online; accessed 18-May-2015].
- [2] [n. d.]. Riscure: Inspector Datasheet. <https://www.riscure.com/security-tools/inspector-sca/>. ([n. d.]).
- [3] [n. d.]. Riscure: VC Glitcher Datasheet. https://www.riscure.com/documents/datasheet_vcglitcher.pdf. ([n. d.]).
- [4] [n. d.]. SAKURA hardware security project. <http://satoh.cs.uec.ac.jp/SAKURA/index.html>. ([n. d.]). [Online; accessed 18-May-2015].
- [5] M Agoyan, Jean-Max Dutertre, David Naccache, Bruno Robisson, and Assia Tria. [n. d.].
- [6] A. Barenghi, L. Breveglieri, I. Koren, and D. Naccache. 2012. Fault Injection Attacks on Cryptographic Devices: Theory, Practice, and Countermeasures. *Proc. IEEE* 100, 11 (Nov 2012), 3056–3076. <https://doi.org/10.1109/JPROC.2012.2188769>
- [7] Jakub Breier, Shivam Bhasin, and Wei He. 2017. An electromagnetic fault injection sensor using Hogge phase-detector. In *Quality Electronic Design (ISQED), 2017 18th International Symposium on*. IEEE, 307–312.
- [8] A. Dehbaoui, J. M. Dutertre, B. Robisson, and A. Tria. 2012. Electromagnetic Transient Faults Injection on a Hardware and a Software Implementations of AES. In *2012 Workshop on Fault Diagnosis and Tolerance in Cryptography*. 7–15. <https://doi.org/10.1109/FDTC.2012.15>
- [9] C. Deshpande, B. Yuce, N. F. Ghalaty, D. Ganta, P. Schaumont, and L. Nazhandali. 2016. A Configurable and Lightweight Timing Monitor for Fault Attack Detection. In *2016 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. 461–466. <https://doi.org/10.1109/ISVLSI.2016.123>
- [10] C. Deshpande, Bilgiday Yuce, Patrick Schaumont, and Leyla Nazhandali. 2017. Employing Dual-complementary Flip-Flops to Detect EMFI Attacks. In *VLSI (AsianHOST), 2017 Asian Hardware Oriented Security and Trust Symposium*. IEEE.
- [11] Naofumi Homma, Yu-ichi Hayashi, Noriyuki Miura, Daisuke Fujimoto, Daichi Tanaka, Makoto Nagata, and Takafumi Aoki. 2014. *EM Attack Is Non-invasive? - Design Methodology and Validity Verification of EM Attack Sensor*. Springer, Berlin, Heidelberg, 1–16. https://doi.org/10.1007/978-3-662-44709-3_1
- [12] Jrn marc Schmidt and Michael Hutter. [n. d.]. Optical and EM Fault-Attacks on CRT-based RSA: Concrete Results. ([n. d.]).
- [13] Noriyuki Miura, Zakaria Najm, Wei He, Shivam Bhasin, Xuan Thuy Ngo, Makoto Nagata, and Jean-Luc Danger. 2016. PLL to the rescue: a novel em fault countermeasure. In *Design Automation Conference (DAC), 2016 53rd ACM/EDAC/IEEE*. IEEE, 1–6.
- [14] N. Moro, A. Dehbaoui, K. Heydemann, B. Robisson, and E. Encrenaz. 2013. Electromagnetic Fault Injection: Towards a Fault Model on a 32-bit Microcontroller. In *2013 Workshop on Fault Diagnosis and Tolerance in Cryptography*. 77–88. <https://doi.org/10.1109/FDTC.2013.9>
- [15] S. Ordas, L. Guillaume-Sage, K. Tobich, J.-M. Dutertre, and P. Maurine. 2015. *Evidence of a Larger EM-Induced Fault Model*. Springer International Publishing, Cham, 245–259. https://doi.org/10.1007/978-3-319-16763-3_15
- [16] Jean-Jacques Quisquater and D Samyde. 2002. Eddy current for magnetic analysis with active sensor. (01 2002).
- [17] S. Skorobogatov. 2009. Local heating attacks on Flash memory devices. In *2009 IEEE International Workshop on Hardware-Oriented Security and Trust*. 1–6. <https://doi.org/10.1109/HST.2009.5225028>
- [18] Sergei P. Skorobogatov. 2005. Semi-invasive attacks – A new approach to hardware security analysis. (2005).
- [19] Sergei P. Skorobogatov and Ross J. Anderson. 2003. Optical Fault Induction Attacks. In *Revised Papers from the 4th International Workshop on Cryptographic Hardware and Embedded Systems (CHES '02)*. Springer-Verlag, London, UK, 2–12. <http://dl.acm.org/citation.cfm?id=648255.752727>
- [20] Loic Zussa, Amine Dehbaoui, Karim Tobich, Jean-Max Dutertre, Philippe Maurine, Ludovic Guillaume-Sage, Jessy Clediere, and Assia Tria. 2014. Efficiency of a glitch detector against electromagnetic fault injection. In *Design, Automation and Test in Europe Conference and Exhibition (DATE), 2014*. IEEE, 1–6.
- [21] Lorc Zussa, Jessy Dutertre, Jean-Max, Bruno Robisson, and Assia Tria. 2012. Investigation of timing constraints violation as a fault injection means. (2012).