Electromagnetic Fault Injection using Transient Pulse Injections

Master's Thesis
A comparison of EM-FI and Optical-FI on smart cards.

Maurice Aarts
m.a.p.aarts@student.tue.nl

1.0

TU/e

Eindhoven, November 2013

Department of Mathematics and Computer Science
Security Research Group

Riscure B.V.

Duration:
March - November 2013

Supervisors:

	I. (Ileana) Buhan-Dulman, PhD	Riscure B.V.
	F. (Federico) Menarini	Riscure B.V.
dr.	J.I. (Jerry) den Hartog	TU/e - W&I - SENS
dr.ir.	L.A.M. (Berry) Schoenmakers	TU/e - W&I - DM

Abstract

An analysis of the effectiveness of fault injection attacks on smart cards using the transient pulse variety of electromagnetic fault injections. This thesis addresses the best testing method to use when using EM-FI technology and describes the types of effects that may be obtained. It also compares the effectiveness of EM-FI in contrast to other methods, such as laser-based optical fault injection, along with the advantages and disadvantages of both. It helps to distinguish in which situations EM-FI is a more viable or better option than optical FI techniques and gives insights into whether common fault injection protection mechanisms and countermeasures are effective against EM-FI or if additional security testing and new EM-FI specific countermeasures and certifications are required.

Preface

I would like to thank, in no apparent order:

- Riscure BV. for allowing me to come to their office in Delft as an intern and use their hardware and resources to do my research.
- Ileana Buhan-Dulman and Federico Menarini, my supervisors at Riscure. Without their support I would not have been able to do this project or bring it to a successful conclusion.
- My colleagues at Riscure, for simply being there to answer all my questions and putting up with me for 6 months.
- My family and friends, for helping me stay focused and pushing me to finish writing this thesis.
- The Kerckhoffs Computer Security Master program, for introducing me to the world of computer security and allowing me to focus on smart card security for my Master Thesis. From Eindhoven University of Technology: my supervisor Jerry den Hartog, and Boris Škorić for the courses on the physical aspects of hardware security and a seminar on RFID/smart card security. From Radboud University in Nijmegen: Lejla Batina and Erik Poll, for the courses they gave that introduced me to side channel attacks, fault injection, and hard- and software security.

About Riscure B.V.

Riscure is an independent security test laboratory specializing in security testing of products based on smart card and embedded technology. Riscure's specialists work with industry leaders worldwide to create products that require strong security to operate safely in a hostile environment. Riscure was amongst the first to apply side channel analysis techniques to smart cards, and pioneered Differential Power Analysis attack techniques.

> "We support security evaluation laboratories, government agencies, manufacturers, and card issuers by conducting security evaluations and by providing and maintaining the Inspector Side Channel Test Tool. Riscure is an EMVco-accredited security evaluation laboratory." [47]

Contents

List of Figures

List of Tables

Listings

Chapter 1

Introduction

1.1 Motivation

Over the past few years, significant advances have been made in the field of computing and embedded devices. Computer systems have become much smaller and more mobile, enabling them to be used in almost every scenario imaginable.

Mobile systems such as smart cards, *Radio Frequency Identification* (RFID) tags and *Systems-on-Chip* (SoC's) have become commonplace in our society. Smart cards, for instance, are used in mobile phone SIM cards, bank cards, passports, identity cards, Pay TVs, physical access systems, and in numerous other applications. With so many different devices collecting, parsing and submitting data it has become more and more important to ensure that such systems are secure. These devices have become safer and more efficient over the years, but the technological advancements have not stopped there. These small mobile and embedded devices are now more than ever the focus of attempts from hackers to obtain (cryptographic) secrets or other information from the device. As it is often the case with security, it has become a race to design new schemes to protect such mobile and embedded devices from the hackers that are developing new ways to circumvent the countermeasures and to obtain access to whatever it is that the device is attempting to protect.

Most smart cards and other embedded devices are, to some degree, sensitive to *side-channel analysis* (SCA) or *fault-injection* (FI) attacks. Currently many of the designers of such systems are developing new and improved countermeasures against the existing types of SCA and FI.

Side channel analysis, often referred to as SCA, is the practice of using unintended leakage of information from a hardware implementation of a program or protocol in order to learn secret information about that program or protocol. All hardware is constrained by the laws of physics, so when a device is doing a computation, it uses a certain amount of power, generates a certain amount of heat, may emit a certain amount of radiation or energy, or may use a certain amount of time to do a specific operation. Often such emanations are correlated to the actual computation that is being done on the hardware. By monitoring a device for such emanations an attacker can use this information to identify weaknesses in the protocol, which in turn can be used to identify the secrets contained within the device. Typical side channels range from the total power consumption and simply timing how long it takes to do an operation, to the monitoring of voltages on individual wires in the circuits, to analyzing the electromagnetic radiation or even monitoring for photon-emission.

Fault Injection, also referred to as simply FI, is the process of influencing a device in such a way that the device starts making computational errors. When an error is introduced, it is called a fault, and when such a fault is successfully injected, it is usually referred to as a "glitch" or a "spike". If the glitch or spike is not detected by the device, it is considered a "successful" glitch or spike; however, many recent devices have countermeasures to detect fault injection attacks and will mute their output, reset the device, or even self-destruct when a fault injection attack is detected.

1.2 Research Goals

One of the newer and lesser known fault injection techniques is *Electromagnetic Fault Injection* (EM-FI). While it is conceptually very similar to optical FI, it has a number of significant advantages with respect to standard optical FI setups. Many current smart card and embedded device implementations have countermeasures against common fault injection attacks, including against optical fault injections. Often these devices are not yet protected against attacks based on transient gradients in the electromagnetic field.

The goal of this thesis is to find an answer for the question:

> How does EM-FI compare to other fault injection methods, such as Optical-FI, with respect to testing techniques and sensitivity to countermeasures?

We want to know what the advantages and disadvantages of EM-FI are with respect to existing FI methods, to help establish the impact EM-FI can have on the state of current fault injection techniques and the ongoing protection against those same fault injection techniques. Smart cards and embedded devices feature a broad range of countermeasures and are often certified with different classifications that show that the chips are protected against certain types of attacks, but little is known about how those countermeasures effect the chips' sensitivity to EM-FI. Is EM-FI the next best thing for attackers and the nightmare of chip producers, or is EM-FI too small in the scope of things to really matter?

The research question can be solved by dividing it into individual points of interest. This leads to the following three sub questions, namely:

1. What testing approach is needed when using EM-FI, and what kind of effects are possible?
2. Are common fault injection protection mechanisms and countermeasures, such as countermeasures against Optical-FI, effective against EM-FI or does the emergence of EM-FI require additional security measures, testing and certifications for smart cards?
3. What are the advantages and disadvantages of EM-FI versus Optical-FI?

Currently there is little information available publicly about the countermeasures on smart cards. This makes it difficult to tell whether a specific card is secure against specific attacks. Security auditing companies, such as Riscure, require a testing methodology or plan to determine which types of test to try on devices that they evaluate, often without knowing the exact countermeasures on the device. We want to help develop a methodology that helps to audit the device under test using an empiric approach. First we will test a selection of smart cards for EM-FI vulnerability, after which we will compare these results for these cards against other fault injection attack methods. This will help to address subquestion 1.

We know what kind of countermeasures are on some of the cards under test, so we can also see if those countermeasures work against EM-FI. This will help to address subquestion 2. Additionally we can compare the tested cards with a known vulnerability to other FI attacks to see if EM-FI is also effective there, and see if perhaps EM-FI is effective on cards that were secure against other types of FI attacks.

Finally we want to give an overview of the advantages and disadvantages of EM-FI with respect to Optical-FI, to answer the last subquestion. The answers to each of these subquestions together should form a sufficient picture to successfully answer the research question.

1.3 Thesis Overview

This thesis gives a broad overview of fault injection attacks and countermeasures, followed by a more in-depth overview of EM-FI.

In Chapter 2 we take a brief look at some of the prior research that has been done on this and related topics in the past.

Chapter 3 contains an overview of what smart cards are and how they communicate with the world.

Chapter 4 gives an introduction to fault-injection techniques. It describes FI in general, after which it focuses specifically on the different types of FI currently available.

Chapter 5 gives an extended overview of the different types of countermeasures that can be used to help prevent SCA and/or FI attacks.

Chapter 6 analyzes the inner workings of EM-FI attacks. It describes the setup commonly used as well as the procedures used to obtain results. This chapter also contains an investigation into the effectiveness of EM-FI by attempting to use an electromagnetic probe on a number of different smart cards both with and without typical countermeasures. These cards are anonymized and are referred to using aliases to prevent direct misuse of the results of the experiments that were done.

Chapter 7 elaborates on the differences between EM-FI and Optical-FI attack methods. It contains an experiment using Laser-FI and a practical comparison between EM-FI and Optical-FI attacks.

Finally in Chapters 8 and 9 we analyze the results of the research in Chapters 6 and 7 and discuss our conclusions as well as point out a few related areas that look interesting for additional research.

The appendices contain a quick overview of any abbreviations, definitions and other notations that may be helpful in the rest of the thesis, as well as a set of full-page sized versions of some of the figures that appear elsewhere in this document.

Chapter 2

Related Work

Smart cards were long considered to be tamper-proof devices, until an article by Ross Anderson and Markus Kuhn [4] suggested that smart cards may at most be tamper-resistant, but definitely not tamper-proof. When Boneh, DeMillo and Lipton released a press advisory about using hardware faults during cryptanalysis [10] in 1996, immediately followed by Anderson's paper about low cost side channel attacks [5], it caused a number of major changes in the way the world looked at the security of smart cards and similar devices. Since then, researchers have developed an entirely new subset of threat models based on the different types of attacks which can be done on the hardware implementations of cryptographic systems.

In the mid 1990's Kocher et al. and Fahn et al. wrote their initial papers about Power Analysis [27] and Inferential Power Analysis (IPA) [17] respectively. Their research into passive attacks initiated the field of side channel analysis, which has been a hot topic ever since. Jean-Jacques Quisquater et al. helped to expand the field of side channel analysis by introducing electromagnetic side-channels [39] in 2001, which in turn led to more research being done into electromagnetic emanations and perturbations. In the following years there were a number of different papers written about both SCA and fault attacks:

- **2004:** Martin Otto wrote his PhD. dissertation on fault attacks and countermeasures [33].
- **2004:** Vincent Carlier et al. analyzed a hardware implementation of AES on a FPGA using differential Electromagnetic analysis (DEMA) [14].
- **2005:** F.Koeune et al. wrote a tutorial which outlines how to go about doing SCA attacks on cryptographic systems [28].
- **2007:** E.Peeters wrote about power analysis and electromagnetic analysis using a new 'switching distance' model instead of Hamming-weight on CMOS devices [36].

In 2007 C.Kim and J.Quisquater did a survey of the different types of fault attacks as well as the fault-injection error-models that can be used. According to them EM-FI is significantly less practical than optical FI but they neglect to explain why [26].

J.Schmidt wrote a paper about DFA in which he used optical (laser) as well as spark-gap-based EM fault injection techniques to break a CRT-RSA implementation on a decapsulated chip [51, 52]. The author does not directly compare optical and electromagnetic fault injection, but does refer to EM-FI as "*the more powerful brother of optical fault induction, as the induced current is much higher.*"

In 2009 M.Akkar et al. developed technology to help improve the smart card development methodology regarding fault-injection attacks by automatically enforcing countermeasures during the development of the smart card's hardware and embedded software layers [29]. This helps to ensure that new smart cards are developed with basic countermeasures to prevent a subset of standard side channels.

Research into the susceptibility of integrated circuits to conducted EMI[1] was done by Ognjen Jović in 2009 [25]. Similarly Ali Alaeldine et al. did research into the influences of changes in the near-electromagnetic field on the operations done by multi-core logic chips without proper EM shielding. Their work shows that unshielded integrated circuits, including ones with EMI protection strategies, are vulnerable to both magnetic and to a greater extent electric field interference [3].

D.Oswald wrote his dissertation [35] about a generic framework for SCA and FI attacks in 2009. Meanwhile A.Sere et al. were writing about the detection of FI attacks and countermeasures [53]. Less than a year later G.Canivet et al. did research into the glitching of secure AES implementations on SRAM-based chips using power glitching and injecting faults with a laser [12].

Hayashi et al. wrote two papers in 2011 on non-invasive intentional electromagnetic interference (IEMI). Their research focused on transmitting a sinusoidal harmonic EM wave through the power cable or an antenna in order to bypass many typical voltage or power-regulating components and similar countermeasures without leaving hard evidence of their attack [19, 20]. In 2012 A.Dehbaoui et al. used transient electromagnetic pulses from a $500\mu m$ and a $1mm$-diameter coil-probe to inject faults into both a software and a hardware based AES implementation [16, 15].

P.Bayon et al. attempted to do an active electromagnetic attack on a ring oscillator based true random number generator (TRNG) with non-invasive methods. Their attempts show that it is possible to lock a TRNG based on 50 ring oscillators on a fixed value based on the injected signal instead of obtaining a truly random value after each attempt, even when using very low power harmonic EM fields. Their research shows that EM interference has significant security implications for random number generators and thus the security of systems that depend on the randomness of such implementations [7].

Woudenberg et al. developed empiric fault injection methods to show that protected smart cards are still vulnerable. They performed power signal guided fault injection, using a triggering mechanism based on real-time pattern recognition and jitter-free diode lasers to show that current countermeasures may be inadequate for the near future [55].

Philippe Maurine did a theoretical comparison of Systems on Chip (SoC) versus smart cards, and describes two possible ways to inject faults into these systems [34].

Sebastian Carlier did research into the optimal shape for an EM-FI probe [13] in 2012. His work was followed up on by an internal project at Riscure B.V. in an unpublished paper [43].

[1]EMI: Electromagnetic Interference

Chapter 3

Smart Cards

A smart card, chip card, or integrated circuit card (ICC) is any pocket-sized card with embedded integrated circuits. Smart cards can provide identification, authentication, data storage and application processing. The smart card is the youngest and cleverest member of the family of identification cards in the ID-1 (ie. credit card) or the ID-000 (ie. SIM card) [24] format. Its characteristic feature is an integrated circuit embedded in the card, which has components for transmitting, storing and processing data. The data can be transmitted using either contacts on the surface of the card, which is called a contact card; or through electromagnetic fields, without any physical contacts, which is called a contactless card [40, 22]. The cards have a tamper-resistant security system, such as a secure cryptoprocessor and a secured file system. Smart cards depend on an external terminal or other card-reading device such as a ticket reader, ATM, or GSM phone to be able to communicate with the rest of the world.

Communication with a smart card can be done in a number of different ways, both through the contact interface and through the contactless interface. Smart cards with a contact interface usually have a contact area with 6 or 8 contacts, as described in ISO 7816 [23]. Each contact has a specific location and purpose. The contacts are placed as shown in Figure 3.1.

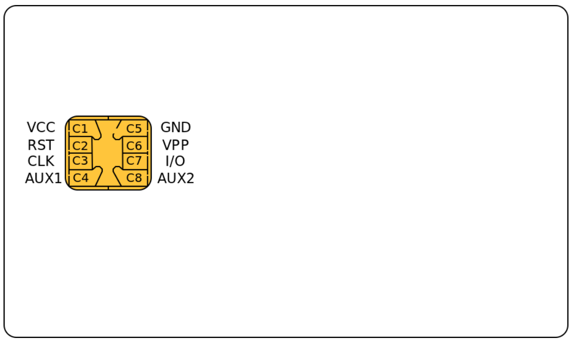


Figure 3.1: A basic smart card with the contact pads labeled.

The **VCC** contact is connected to the VCC line of the embedded chip and provides the power for the chip. **RST** is the reset contact, which forces the chip to reset when it is triggered. The **CLK** pad allows a chip to make use of an external signal for data signal timing and internal operations. Some cards have an internal clock for the chip itself and only use the external clock pulses for data transmission synchronization. **GND** is the ground pad, it connects the chip to the reference voltage supplied by the card reader. The **VPP** pad is a relic of the first generation of smart cards and memory cards. The pad was used to allow the reader to supply a specific voltage to be used for programming the EEPROM of the chip, as this usually requires a higher voltage than the chip itself. Almost all recent implementations use a charge-pump so that they can manage their own programming voltages and so that they are less vulnerable to tampering. According to ISO 7816-3:2006 [23] the **VPP** pad is now designated for either standard or proprietary use, meaning that it is also allowed to use the **VPP** pad as an additional I/O pad. The **I/O** pad is a serial connection to the chip supporting half-duplex input and output. The remaining two pads, **AUX1** and **AUX2**, are usually either not available, available but not connected to anything, or used for a USB interface with the chip.

There are typically three classes of smart cards, classified based on the VCC voltage that they use; Class A is the broadest class of smart cards, but the newer classes, such as Class C cards are becoming more popular as the use of low power devices such as mobile phones increases due to advancements in chip die-size causing the chips to require less power and consumers requiring longer battery life for their devices. The typical voltages for the different classes of devices are given in Table 3.1.

Class A	$5V \pm 10\%$	1-5 MHz
Class B	$3V \pm 10\%$	1-5 MHz
Class C	$1.8V \pm 10\%$	1-5 MHz

Table 3.1: The basic classes of smart cards.

3.1 Communication

Smart cards typically start their communications by sending an Answer to Reset message, or ATR. This happens as soon as the chip is powered by a reader and the chip has finished its booting process. This ATR signals that it is ready to start receiving commands from the reader. The ATR contains a large amount of information about the smart card and the chip being used. The ATR contains information about the communication parameters proposed by the card, the card's nature, and its state. Almost all smart cards use asynchronous communications while most memory cards use synchronous transmission. The ATR typically depends on the type of transmission used; for the asynchronous transmission variant the ATR is extensively normalized and defined by the ISO/IEC 7816-3 [23] standard. This allows it to be used to identify a large number of properties that the smart card requests when it is to be used. After the card sends the ATR the reader selects a protocol supported by the smart card and then starts transmitting commands to the smart card.

3.1.1 ATR - Asynchronous Transmission

The ATR starts with an initial byte TS, followed by a maximum of 32 characters. The TS byte encodes the convention used for the communications until the next reset. This byte is typically either `0x(`3B`)` for direct convention and `0x3F` for inverse-convention respectively. The TS byte is followed by a $T0$ byte which encodes the format for the next part of the message.

The $T0$ byte consists of 4 low-order bits and 4 high-order bits. The low-order bits encode the number of historical bytes, in the range of [0..15]. The high-order bits of $T0$ encode the presence of the interface bytes TA_1, TB_1, TC_1 and TD_1 by setting a value of 1 for the 5^{th}, 6^{th}, 7^{th} and 8^{th} bits respectively. The $T0$ byte is followed by the interface bytes TA_1, TB_1, TC_1 and TD_1 as defined by the $T0$ byte. These bytes are optional and encode the communication parameters that the card supports. As an example, a common value for the $T0$ byte is `0xF8`. This indicates that

TA_1, TB_1, TC_1 and TD_1 are present, and that they are followed by 8 historical bytes. Similarly `0x65` indicates that only the TB_1 and TC_1 bytes are present, along with 5 bytes of historical data.

The interface bytes are used to encode the protocols the card supports, the maximal clock frequency supported, the number of clock periods per ETU[1], global properties such as the **VPP** voltage, additional guard time[2], or even additional blocks of interface bytes. The interface bytes are followed by a set of historical bytes. The historical bytes T_i describe the operating characteristics of the card using at most 15 bytes of data. This data can be normal ASCII, or it can be formated as a TLV *(Tag, Length, Value)* object in accordance with ISO 7816-4 [23].

The last byte of the ATR, if present, is the Check byte (TCK). If the only indicated protocol in TA_i is $T = 0$ then the TCK byte will not be present, but in all other cases the TCK is meant to add redundancy to the ATR. When the TCK byte is present, XOR-ing all the bytes from $T0$ to TCK (inclusive) should result in the value `0x00`. If this is not the case then the ATR is invalid.

3.1.2 ATR - Synchronous Transmission

The synchronous transmission ATR is defined by the ISO/IEC 7816-10 [23] standard, but generally those ATRs exist of 32 bits, organized into 4 bytes. The first byte, denoted H1, codes the protocol the smart card wishes to use, with `0x00` and `0xFF` defined as invalid codes. The second byte of the ATR, denoted H2, codes for the parameters of the protocol. Bytes 3 and 4, H3 and H4 respectively are not standardized and thus their contents can differ from card to card.

3.1.3 Attacks

Basically, there are four classes of attacks: Reverse engineering of the hardware, fault attacks, side-channel attacks, and software attacks. On top of this there are attacks that combine elements out of these four fundamental classes [56]. Fault attacks are typically much less invasive than reverse engineering, and are performed with the aim to introduce faults either during code execution, or when reading data or code from the various memories.

A famous example of a fault attack is the Bellcore [11] attack, where the introduction of a single fault at the right stage of an RSA calculation based on the Chinese Remainder Theorem will reveal the secret key used. The most economical way to address these attacks is with a right mix of sufficiently resilient hardware and a robust embedded operating system that can cope with "mistakes" made by the hardware. The Bellcore attack already demonstrates a key aspect of these types of attacks: Often, it suffices to have only a few successful hits in order to succeed with the attack. However, the embedded software has a chance of detecting the attack, e.g., when doing redundancy checks that go wrong, or by monitoring alarms sent from the underlying hardware.

Some fault attacks like safe-error-attacks [31] are rather difficult to cope with, though, since there the exploit already happens by detecting an "unusual" response to the attack by the embedded device - e.g., a reset forced by an attack is already a useful information [56].

[1]ETU: Elementary Time Unit. The duration of bits. Typically one third of the delay between the first and second H-to-L transition in the TS byte. This is optional, as the ATR standard defines a default of 372 periods of the external clock signal, available from the reader. $R = {}^{372}/_{1}$

[2]The guard time is the delay that the card requires following the leading edge of the previous character transmitted to or from the card.

Chapter 4

Fault Injection

4.1 Introduction to Fault Injection

Fault injection is the process of influencing a device in such a way that the device starts making computational errors. When an error is introduced, it is called a fault, and when such a fault is successfully injected, it is usually referred to as a “glitch” or a “spike”. If the glitch or spike is not detected by the device, it is considered a “successful” glitch or spike; however, many recent devices have countermeasures to detect fault injection attacks and will mute their output, reset the device, or even self-destruct when a fault attack is detected.

Simply causing an error on a device is usually not sufficient to obtain secret information about the device, so faults must be introduced at specific points during a device’s calculations, so that the output of the device is affected by the fault injection. Faults can be used to affect the value of a single bit (a bit flip fault) or by changing the entire flow of the program being executed. By injecting faults repeatedly and analyzing the output it becomes possible to identify the relation between the timing of the fault injection and the output of the program. Upon further analysis it is often possible to use these faults to read secret information from memory that should not be readable. Another possibility is to time a fault injection in such a way that the result of a check is modified such that the result is considered valid instead of invalid.

```
 1 void verify_pin(unsigned char *input) {
 2     static const char pin[4] = {1,3,3,7}; // static pincode (secret)
 3     volatile char digits_ok = 0;          // pincode digit counter
 4     uint8_t i;                            // for loop counter
 5     for (i=0;i<4;i++) {
 6         // Check 4 digit pincode, digit by digit
 7         if (pin[i] == input[i]){
 8             digits_ok++;
 9         }
10     }
11     if (digits_ok==4) {
12         // Command successfully executed, send 0x9000
13         respond_code(0x00,SW_NO_ERROR_msb,SW_NO_ERROR_lsb);
14     } else {
15         // Usage conditions not satisfied, send 0x6985
16         respond_code(0x00,0x69,0x85);
17     }
18 }
```

Listing 4.1: A simple pincode check implementation. (Source: [48])

The code snippet in Listing 4.1 contains the java code for an extremely simple pincode check against a hard coded pincode that could be implemented on a smart card. Line 5 of Listing 4.1

contains a loop that checks each digit in the *input* against the correct digit from the stored *pin*. If we inject a fault such that the check on line 7 returns *true* regardless of whether the first *input* digit actually matched the pincode's first digit, and repeat that glitch for each of the remaining three digits, we can make the card think that we entered a valid pincode and successfully authenticated without ever knowing the actual pincode. If we look a little bit further, we see a check on line 11 where we can achieve the same result with only a single glitch instead of four.

We could also attempt to modify the value of one of the variables that is being read from memory. If we use side channel analysis to monitor the value of the registers, we might find where the variable *digits_ok* is stored. We then might be able to use fault injection somewhere between lines 1 and 11 on that part of the device to increment the value of the *digits_ok* variable to the value 4. This would allow the check on line 11 to return true even if the actual pincode check failed. In a similar fashion we might find the register containing the variable *i*, and we could modify it to a fixed value during the loop, so that it would check a specific index each time the check on line 7 is evaluated, and then increment it to a value greater than 4 so that the loop terminates. This would allow us to input a pincode such as 1111 and have it authenticate against the secret pincode of 1337 by fixing the value of *i* to 1 for each recursion of the loop, and then set *i* to a value greater than 4 to terminate the loop.

The best technique depends heavily on the code used, as well as the type of fault injection which is being attempted. For the first example, where a check needs to be skipped, it is often enough to simply raise or lower the voltage on the VCC line of the device. For the register and memory based fault injections, a much finer control of the glitches is often required as individual bits need to be flipped, or only certain bus line values must be changed.

Fault injection attacks can also be used in combination with side channel analysis. By introducing a large number of small faults in repeated executions of an algorithm, it becomes possible to analyze the results and find a correlation between the faults and the results that are obtained. This type of fault attack is called a Differential Fault Analysis (DFA) attack [51].

4.2 Injecting Faults

Glitching is a manner of injecting faults on a device either through physical contact with the device or by modifying the environment in such a way so that it has an effect on the device. There are three basic ways of glitching an electrical device, and all three are based on increasing or decreasing the voltage on a certain wire in the device.

Figure 4.1 shows a number of parameters that can be changed for certain types of glitches. For instance, the V_{CC} voltage can be changed prior to or during a glitch to limit the power consumption of the smart card. Lowering the V_{CC} below the threshold values will often disable certain countermeasures or make it easier to do a successful fault injection. We can also modify the clock high voltage and the clock low voltage, making it easier to cause a glitch in the clock signal by decreasing the difference between the high and low clock signals.

When doing a fault attack using glitches, the first thing we do is to send a command to the card to start doing an interesting computation. We do not always want to send our glitch immediately after starting the computation, so we can wait for an arbitrary number of clock cycles before starting our glitch. This waiting period is referred to as the number of *wait cycles*. We can then choose how often we want to repeat the glitch, using a parameter which is known as the number of *glitch cycles*. Each glitch cycle can have its own number of wait cycles, allowing us to glitch in either two consecutive cycles or to wait for a length of time before doing a second glitch. Within each glitch we have an additional set of parameters which can be modified. These parameters include the length of the glitch, (*glitch length*), the *glitch voltage* or amplitude of the spike that is induced in the signal, and the *glitch offset* which defines the exact timing for when the glitch should start within the respective clock cycle.

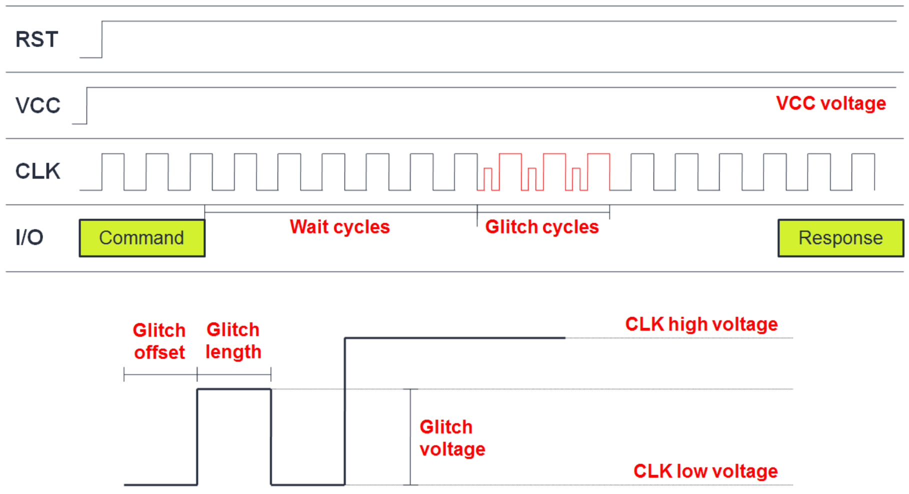


Figure 4.1: An overview of the configurable parameters that can be used when injecting faults into a device under test. [48]

4.2.1 Voltage Glitching

Voltage glitching, also known as power glitching, is the most basic type of glitching. A voltage glitch is done by simply raising or lowering the voltage of a targeted wire in the device for a fixed period of time. According to ISO7816-3 [23] smart cards must tolerate a certain variation in the power supply V_{CC} of $\pm 10\%$ of the standard supply voltage of 5V. If the variation is significantly higher than 10% the card is no longer required to work properly. Short variations in the power supply, known as *spikes*, can be used to induce faults in the smart card's computation. Spikes can be used to cause faults in the smart card's memory as well as to modify the program execution and code flow. A spike can affect a single bit, but may also be used to change an arbitrary number of bits. Spikes can increment or decrement values used in the program, such as loop counters or flags, and they can also change the evaluation of conditional statements. Spikes can have different effects based on nine parameters including timing, spike length, voltage value, and the transition shape [33].

4.2.2 Clock Glitching

Clock glitching is a fault injection technique that relies on modifications of the smart card's clock signal. Many older smart cards do not have an internal clock and thus rely on an external clock signal from the card reader terminal. More recent smart cards may use a randomized clock, by randomizing the clock signal provided by the card reader. New smart cards often have an internal clock for the actual processing and use the external clock signal only to synchronize the communications with the card reader. Smart cards are required to tolerate a clock voltage variation from 0 to $0.5 \cdot V_{CC}$ for the low clock signal and $0.7 \cdot V_{CC}$ to V_{CC} for the high clock signal. They must also tolerate variations of up to 9% in the rise and fall times for the clock signal [33]. Figure 4.2 shows a setup that can be used for both voltage and clock glitching on smart cards.

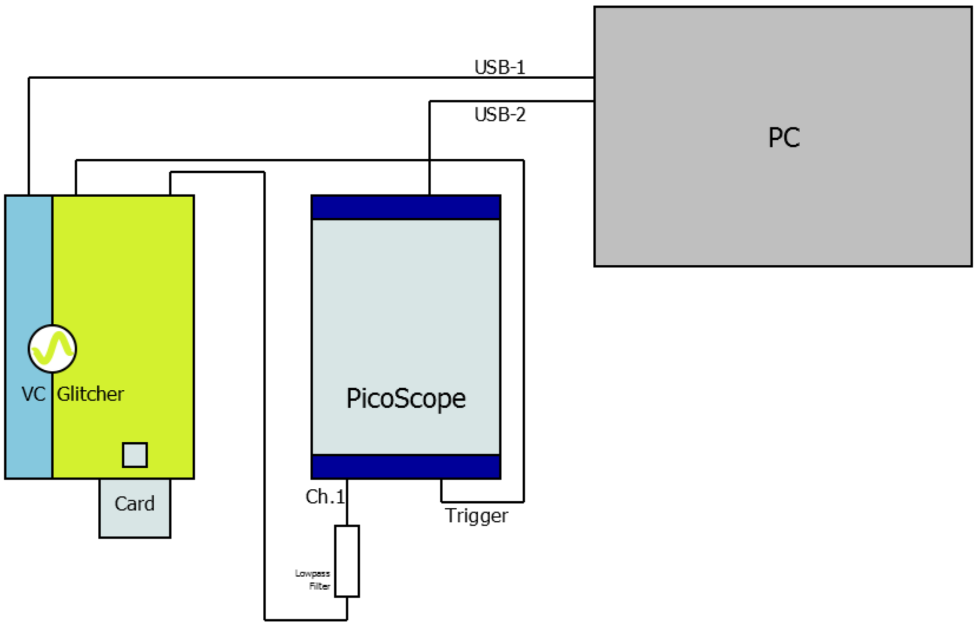


Figure 4.2: A standard configuration for voltage or clock based fault injection on smart cards using the VC Glitcher, a Picoscope, and a low-pass filter.

4.2.3 Harmonic Glitching

Harmonics are sinusoidal voltages or currents having frequencies that are whole multiples of the frequency at which the supply system is designed to operate. By injecting a signal that is a harmonic of the actual signal it is possible to adjust the functioning of the device. An example of such a technique is modifying the oscillation of a random number generator by using harmonics to synchronize the oscillations. In doing so the randomness of the RNG becomes predictable and thus the entire purpose of the RNG is defeated.

4.3 Optical Fault Injection

Optical fault injection almost always requires for the chip or embedded device to be decapped, so that the silicon parts of the chip become visible. This is necessary to allow the light from the laser to reach the chips surface. Only in rare cases, for example when a transparent epoxy is used, is it not necessary to decap the chips. Figure 4.3 shows a GSM smart card chip in a decapped state. In this case the decapping was done using a combination of a scalpel and corrosive chemicals such as nitric acid (HNO_3), but purely mechanical as well as optical methods of decapping are also possible.

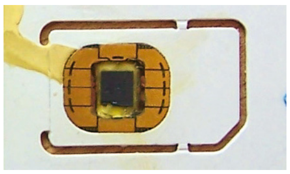


Figure 4.3: A decapped smart card chip in GSM (ID000) format [42].

A green laser (532 nm wavelength) or red laser (808 nm wavelength) is designed for frontside testing of smart card chips and embedded devices. In combination with optics it is capable of producing a spot size of 6 x 1.4 μm on the chip surface. This gives an accurate control over the chip area. The laser has sufficient power to penetrate through the gaps in the shielding commonly applied in today's secure chips (see Figure 4.4). The near-infrared laser (1064 nm wavelength) is designed for backside testing of smart card chips. This powerful laser penetrates the chip substrate to reach the transistors [42].

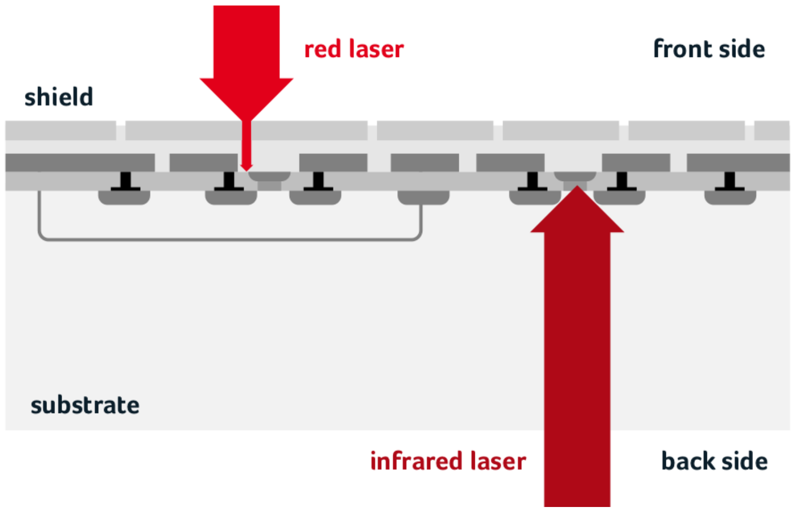


Figure 4.4: The effect of a laser's beam on a chip [42].

Smart card chips differ and to identify a chip's weakest spot, one has to be able to accurately adjust the strength of the laser pulse. Figure 4.5 shows the relationship between laser energy and the effect on an integrated circuit. When injecting too little energy, there is no effect, and when injecting too much energy, the so-called latch-up effect occurs which causes chip damage. Only when injecting the right amount of energy, integrated circuitry can be effectively manipulated [42].

The laser's photons generate free electrons in the P- and N-channel of transistors. As a result the conductivity of any transistor inside the laser spot increases and transistors switch to *On* state. Conductance of both transistors of a P- and N-channel pair causes short circuiting between V_{DD} and GND which may damage the chip. The laser located above *M2* and *M1* in Figure 4.6 changes

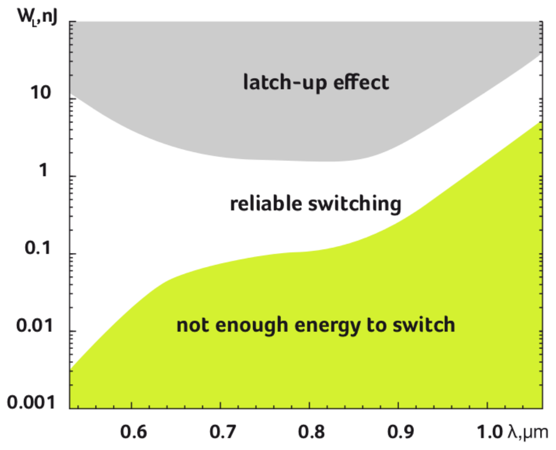


Figure 4.5: The relationship between injected laser energy and effect on transistors. [42].

the status of *M1* or *M2* or both, which depends on laser's wavelength and pulse strength. On a lower level, rather than short circuiting field-effect transistors as lasers the faults induced with EM-FI causes the flux to change in a particular part of a circuit changing the state of the field-effect transistors. This behavior has also proven to be less destructive for devices saving precious samples and work while performing a testing scenario [50].

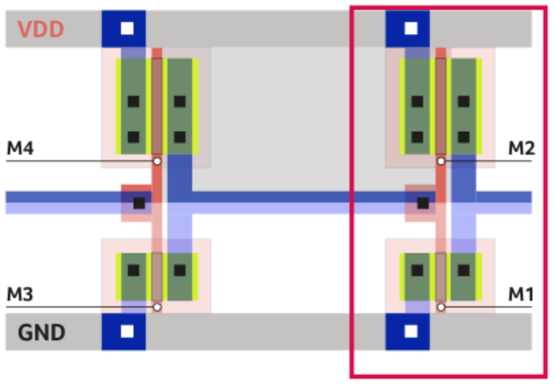


Figure 4.6: The effect of a laser's pulse on a chip's circuit [50].

4.4 Electromagnetic Fault Injection

Electromagnetic fault injection is based on electromagnetic induction. Electromagnetic induction is caused by the interaction between electric fields and magnetic fields, according to Ampére's circuital law and the Maxwell-Faraday equations. The process of alternating the polarity of a magnetic field in time causes an electric field to be generated. The consequence of this is if a varying magnetic field flux passes through a closed circuit, then a current is generated in that circuit [43] . Figure 4.7 shows a schematic view of the effect of a magnetic field on a ring-shaped conductor. If the strength of the flux (black arrows in the figure) is changed, then the current in the ring (white arrows) will increase or decrease with respect to the changes in the magnetic flux. If the polarity of the flux is changed then the direction of the current is switched as well. Similarly a magnetic field can be generated by varying the current in a ring-shaped conductor.

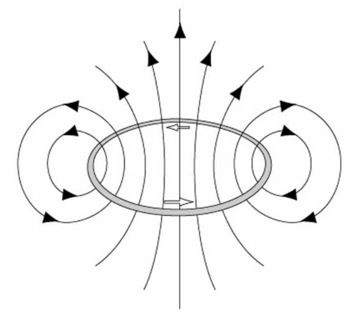


Figure 4.7: A schematic view of a loop of electrical wire with a current in a counterclockwise direction. The current generates a magnetic field as shown by the black arrows. [43]

The intensity of the magnetic field, B, in a ring-shaped conductor with radius R and constant current I in vacuum can be calculated using Equation 4.1.

$$B = \frac{\mu_0 \mu_r I}{4\pi R^2} \oint dL = \frac{\mu_0 \mu_r I}{4\pi R^2} 2\pi R = \frac{\mu_0 \mu_r I}{2R} \tag{4.1}$$

The equation contains a constant μ_r which represents the magnetic permeability of the material that the magnetic field is passing through. In vacuum this constant is 1, but in other materials, such as air it is 1.2566375×10^6 or for ferrite it ranges from 2.0×10^5 to 8.0×10^4] depending on the nickel to zinc ratio. The current induced in a loop by a non-stationary magnetic field can be calculated using the Maxwell-Faraday Equation 4.2.

$$\oint_{\delta \sum} E \cdot dl = -\frac{d}{dt} \iint_{\sum} B \cdot dS \tag{4.2}$$

This equation links the variation in time of the magnetic field B across a surface $\sum$, to the electric field E generated on the border l of the surface $\sum$. This equation implies that an electric field can be injected into a closed circuit by simply varying the B field traversing the surface identified by the circuit itself [43]. This electric field also generates a difference of potentials across different parts of the chip itself, which in turn induces currents as the free charges move to follow the potential gradient. The strength of the induced potential gradient is dependent on the time-derivative of the magnetic flux, and not the absolute value of the flux itself. This can be seen in Equation 4.3. This means that stronger currents and potential gradients can be induced by faster and sharper variations of the magnetic field $B(t)$.

$$\frac{dB(t)}{dt} = \frac{d\left(\frac{\mu_0 \mu_r I(t)}{2r}\right)}{dt} = \left(\frac{\mu_0 \mu_r}{2r}\right)\left(\frac{dI(t)}{dt}\right) \tag{4.3}$$

The current $I(t)$ in the wire loop can be generated by varying the voltage V as a function of the time. If we assume that the wire has a resistivity R, we obtain the expression in Equation 4.4.

$$\frac{dB}{dt} = \left(\frac{\mu_0 \mu_r}{2r}\right)\left(\frac{1}{R}\right)\left(\frac{dV(t)}{dt}\right) \tag{4.4}$$

If the changes in voltage are done linearly, then $\frac{dV(t)}{dt} = m$ gives a linear term, which shows that the magnetic field in Equation 4.4 changes linearly as well.

In order to use these effects to do fault injections on a smart card, we intend to use the effects above twice. We need to place the smart card chip in a strong magnetic field that can be varied enough to cause a fault in the chips circuits. As we can use a loop with a varying current to induce a magnetic field, and we can use that magnetic field to induce a current in a ring-shaped conductor, we can also create a device that can be used to generate magnetic fields which in turn induce currents in the smart card's chip. Figure 4.8 shows a schematic of such a device.

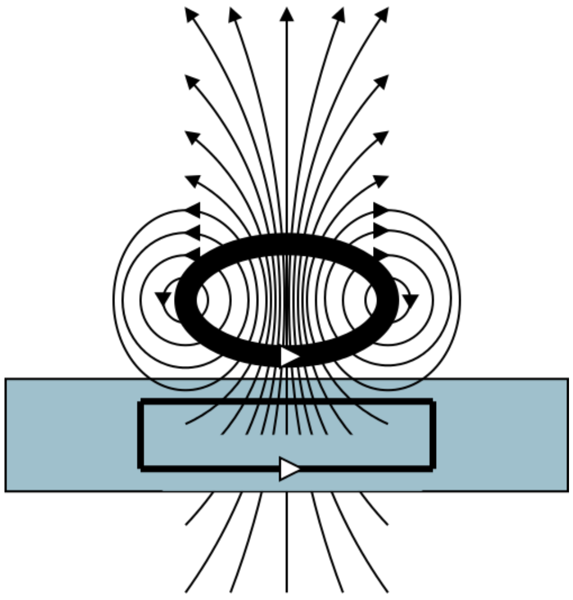


Figure 4.8: A schematic view of a electromagnetic fault injection probe. The ring is charged with a variable current, which produces a magnetic field, which in turn induces a current in the (rectangular) circuit in the smart card or device under test. The arrowheads represent the direction of the magnetic field (black) and the current (white).

When we place a smart card in the magnetic field of the EM-FI probe, we must position it in such a way that the magnetic field has the most effect on the circuits in the chip. In order to get the most effect, the electromagnetic pulse should hit the "active side" of the embedded chip. Usually the embedded chip is mounted with the active side of the chip facing the backside of the smart card. As the chip is mounted backwards with respect to the smart card itself, the "frontside" or "active side" of the chip is on the (plain) backside of the smart card, and the "backside" of the chip is on the frontside of the smart card, usually covered by the contact pads if it is a contacted smart card. Figure 4.9 shows the internal components of a smart card for reference, the card's contact pads are shown on top in the figure; and the bottom of the figure shows the backside of the smart card.

When using EM-FI on a smart card, the flux must pass through an element which is vulnerable to magnetic fields. This will usually be the chip itself, or optionally the bond-wires or even the contact pads. In order to get the best result, the magnetic field should be as strong as possible in

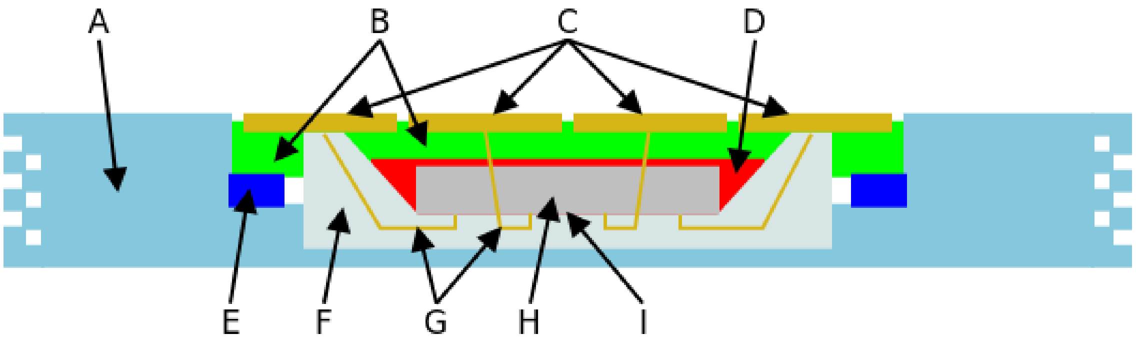


Figure 4.9: A cutaway side view of the components of a smart card. Key: A: card body, B: substrate, C: metal contacts, D: chip adhesive, E: hotmelt adhesive, F: encapsulation, G: bondwires, H: chip, I: active chip surface.

the vulnerable area. As we will usually attempt to do fault injections on the chip, we will position the EM-FI probe on the backside of the smart card, centered above the frontside of the embedded chip. This is done because the contact pads on the frontside of the smart card effectively shield the chip from a considerable amount of magnetic flux. By positioning the probe on the frontside of the chip, the flux will simply pass through the outer plastic card and the epoxy encapsulation layers and have a greater effect on the chip.

There are two basic variants of fault injection using electromagnetic radiation. The first is Transient Pulse EM-FI, which is based on sending a voltage spike through the coil, which causes a jump in the magnetic field, which then causes a current spike to be generated in the target area. The second type of EM-FI is called Harmonic Emission. These are sinusoidal voltages that are induced in the chip such that they do not trigger voltage sensors or similar countermeasures because they appear identical to the normal signal. By modulating the frequency of the induced signal the operations of the chip circuit can be modified.

4.4.1 Transient Pulse EM-FI

A transient electromagnetic pulse is done by briefly alternating the charge of the EM coil with a high voltage to create a strong gradient in the electromagnetic field and thus induce a spike in the device under test. The spike generated is plotted in Figure 4.10, however when the pulse is measured with an oscilloscope it appears as Figure 4.11. This oscillation is an artifact, caused by the oscillation between the ground and the probe itself, the actual signal sent is as shown in Figure 4.10.

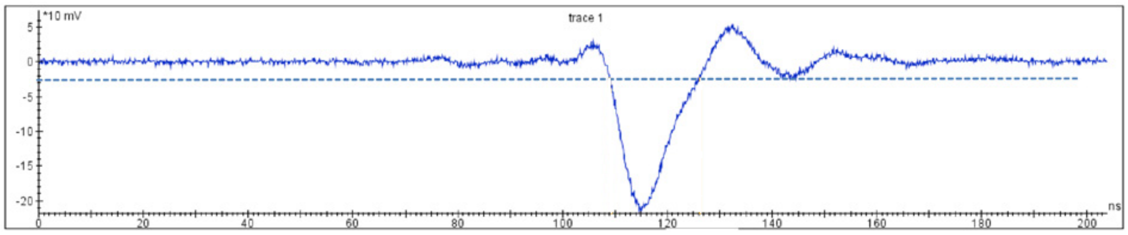


Figure 4.10: A plot showing the characteristic waveform of a single pulse from the EM Coil with a low pass filter [50].

The magnetic pulse induces a voltage glitch in any circuit loop under the coil. The voltage glitch may change a transistor status from *Off* to *On* or vice versa depending of the polarity of the

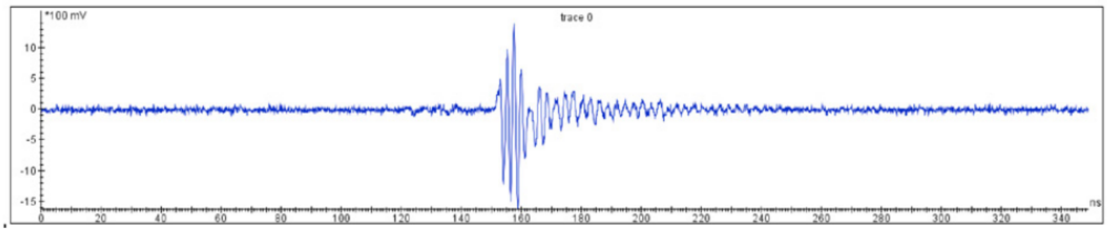


Figure 4.11: A plot showing the characteristic waveform of a single pulse from the EM Coil without a low pass filter. The oscillation shown is a measurement artifact, it is not present in the actual pulse [50].

voltage glitch and type of transistor. The voltage glitch will only switch one transistor of a P- and N-channel transistor pair to *On* and therefore does not cause short circuit between V_{DD} and GND. This is called a Single Event Upset [1]. Figure 4.12 gives an example of a circuit lay-out. The coil located above *M2* (coil not shown in the figure) induces voltage glitches separately in two loops: the loop outlined in red (center) and the loop outlined in orange (far right). Since the loop area in red is much bigger than that in orange, the loop in red will get most effective glitch [50].

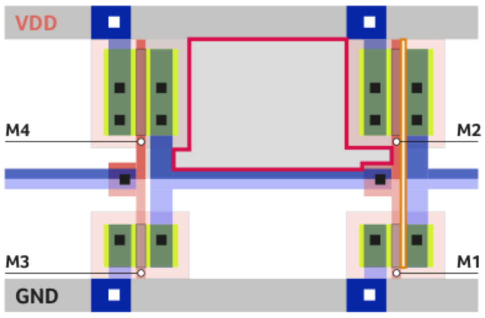


Figure 4.12: The effect of an electromagnetic pulse on a chip's circuit [50].

Often multiple glitches are required to flip multiple bits in a sequence or to repeatedly flip a transistor during a loop. Such a sequence of glitches appears as shown in Figure 4.13. In that figure 50 pulses are sent to the device, after which there is a delay followed by another set of 50 pulses. The bottom plot in the figure shows the voltage of the EM Probe's coil, while the top plot in the figure shows the power trace of the device along with the spikes that the EM Probe has generated.

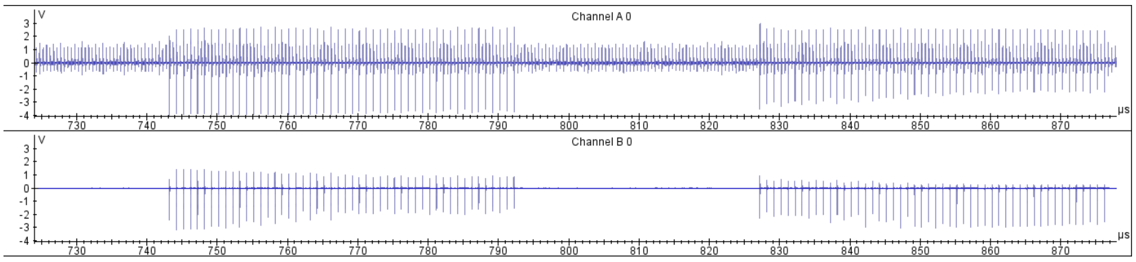


Figure 4.13: A power trace of the response of a smart card to the glitches introduced through the EM Coil. Channel A is the power trace, Channel B is a trace of the voltage over the EM Coil. Here two sets of 50 pulses were sent.

4.4.2 Harmonic Emission EM-FI

Harmonic Emissions EM-FI uses electromagnetic fields to induce harmonic currents in a chip, as described in Section 4.2.3 without having to contact the chip, and thus evading possible countermeasures against VCC and clock glitching. The a typical setup used to do harmonic EM-FI is shown in Figure 4.14.

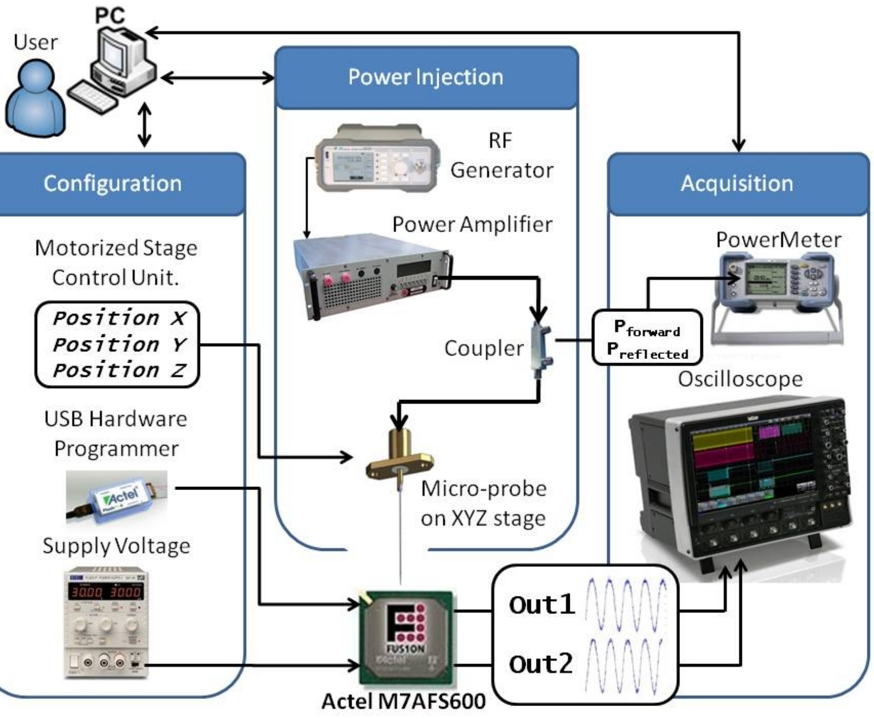


Figure 4.14: A typical setup for Harmonic EM-FI [7].

The probe for this type of EM-FI setup is different from the one used for transient pulse EM-FI. Figure 4.15 shows the probe used. It consists of a 30 mm tungsten rod which has a diameter of 200 μm at one end leading down to a 10 μm tip. This probe involves predominantly electric field at the tip end, which can couple with the metal tracks inside the chip [7].

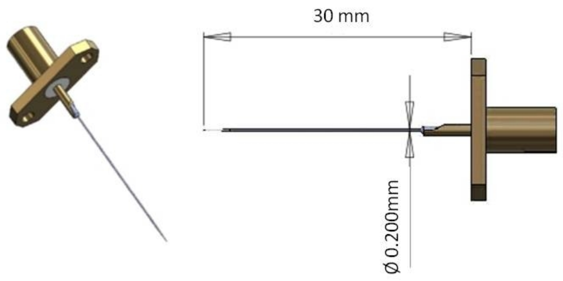


Figure 4.15: A unipole micro-probe for Harmonic EM-FI [7].

Harmonic EM-FI is currently a hot topic and the subject of much ongoing research. More information can be found in [38] and [7].

Chapter 5

Countermeasures

5.1 Introduction to Countermeasures

Smart cards and embedded devices are often protected against side channel attacks and fault injections by using specific countermeasures. These countermeasures can be general, meaning that they work for a range of attacks, or they can be very specific, specialized to keep one certain type of attacker away.

5.2 General Countermeasures

Generally countermeasures can work on three essential levels. The first of these levels is Tamper resistance, or making it exceedingly difficult to access the hardware. The second method is through Tamper response, or by detecting tampering and actively responding to the threat. The third method is through Tamper evidence. This level does not really restrict tampering at that point, but it does help to show that tampering has taken place, making it difficult for an attacker to deny they attacked the device. Each of these three methods is explained in the following sections[1] Note that these countermeasures are general, some methods will not work for certain targets and some methods must be adapted to the specific use case.

5.2.1 Tamper resistance

Tamper resistance relies on restricting physical access to the smart card or embedded device, such that the only interaction has to be done through the software embedded on the device. Of all security methods, tamper resistant security is usually the easiest to apply, as tamper resistant systems usually take the so-called bank vault approach and ensconce the microchip in a protective cover that protects it against invasive attacks[57].

There are many different ways to restrict physical access to an embedded device. Below we have a list of such methods[2], each with a brief description of what the method details and the types of attacks it helps protect against.

[1]These three sections are based on *Hardware Attacks, Tamper Resistance, Tamper Response and Tamper Evidence* [2]
[2]This list is based mainly on [2], which in turn is based on Weingart's work in *Physical Security Devices for Computer Subsystems: A survey of Attacks and Defenses* [57]

“Bank vault” technology By simply making the embedded device too big or heavy to steal can significantly decrease the probability of an attacker stealing the device. The device can also be permanently attached to an object such that the embedded device is destroyed before it can be detached from the object. Note that this is not very convenient for portable devices and thus other technologies have been developed.

Hard Barriers An actual hard physical barrier surrounding the device. Materials such as steel, ceramics, hard plastics and cement or brick can help prevent invasive tampering, and may also prevent theft in combination with the technology above. An example of a hard physical barrier is shown in Figure 5.1.

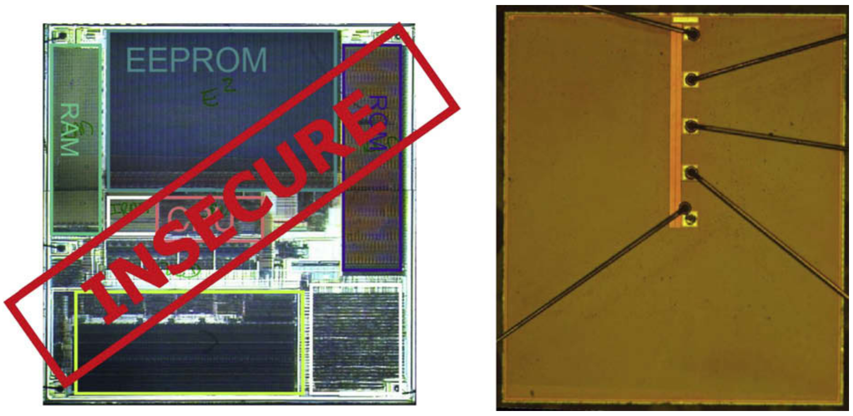


Figure 5.1: Insecure and tamper resistant chips[30]

Metal Shielding Enclosing the device in a metallic cage helps protect it against electromagnetic fields, and embedding layers of metal in the circuit board help obfuscate which traces in the board are causing the magnetic field[30].

Insulator based substrates Silicon becomes transparent to infrared radiation, so in order to prevent against IR laser attacks it is possible to replace the majority of the silicon in the device with a material that is not transparent to IR lasers or other frequencies that enable imaging of the circuits. Some examples of such materials are SiMOX (Silicon/Metal Oxide) and SOS (Silicon-on-Sapphire). Using an insulator based substrate in combination with advanced passivation gives the highest level of passive, single-chip, protection. Note that material machining techniques can still disable this type of security by removing or thinning the substrates to a thickness where the material is too thin to block IR light and allows imaging attacks to take place.

Semiconductor Topography Design By designing the chip in a certain way, or using glue-logic[3] [54] as shown in Figure 5.2, it is possible to ensure that the layers required for functionality surround the layers that need to be kept secret. This ensures that the secret areas cannot be exposed without removing or damaging the functional layers that are required to read the secret data. This technique can be used against pico-probing, scanning electron microscopes and the various machining techniques.

[3]Glue-logic is the logic required to interface circuit modules. The modules are typically complex chips such as microcontrollers, RAM, and peripheral ICs. It helps to obfuscate the individual building blocks by meshing everything together.

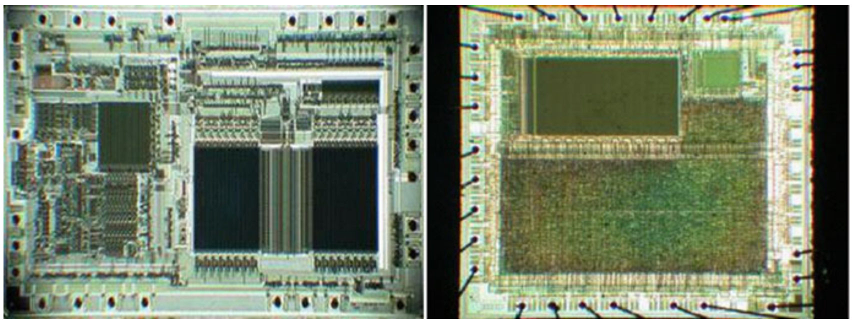


Figure 5.2: A chip with the regular building blocks (left) and a chip with glue logic (right) [54].

Hiding - Noise Generation One way to decrease the signal-to-noise ratio (SNR) is to increase the noise. A number of researchers have proposed adding noise generators to secure ICs in order to foil side-channel attacks. For power analysis attacks, designers have proposed adding circuits that draw random amounts of power during chip operation or that keep the total power dissipation constant by filling in power dissipation if it is lower than a set amount. For timing attacks, researchers have proposed adding circuits that have a random delay into the logic path. However, adding this type of circuit may pose a problem for synchronous systems, which must complete all logic paths by the end of the clock period.

Randomized delays will also make any side-channel observation that relies on sub-sampling (e.g., optical emission) or trace alignment more difficult. For EM attacks, EM noise can be added to reduce the SNR. The difficulty is that the emission spectrum is quite broad required a large amount of added noise across a wide frequency range. Such an EM noisy chip poses a problem for system integrators whose designs must pass governmental tests for electronic interference. Further, this would require a significant amount of power, if not die area, for the noise generators. Finally, unless the noise generators are carefully placed, they may not add noise to all of the unintentional EM emissions due to signal coupling.

While adding noise to the side-channel can be sufficient to make simple side-channel attacks infeasible, differential attacks will still be possible, but require more traces or more advanced signal processing [32].

5.2.2 Tamper response

Whereas tamper resistant systems used a bank vault approach, tamper response systems are more like a burglar alarm. These systems specialize in detecting an intrusion, and if such a detection takes place the chip will instantly attempt to stop the attacker from learning anything else about the system. Such responses can vary from simply sounding an alarm, to clearing the ROMs, to destroying the physical device itself.

Tamper response technology consists of two important parts, the first is detection of an attack, and the second is the actual response if an attack is detected. Detection of an attack can be done by installing sensors on the embedded device. In Steve Weingart's paper *Physical Security Devices for Computer Subsystems: A survey of Attacks and Defenses*[57], he describes a complete list of sensors that can be used to detect a multitude of attacks. The exact shape and type of sensor depends on what it is built to detect, but regardless of the type of sensor it gives an output when an attack is detected. Such an output is caught by the logic that handles the response part of the tamper-response module. These mechanisms fall mainly into four groups:

- Switches - devices that detect mechanical movement.
- Sensors - devices that detect an environmental change.

- Circuitry - wires and/or fiber-optics that are wrapped around and throughout the embedded device. These materials are used to detect a break, puncture or attempted modification of the wrapper[18].
- Electronic - detection and monitoring of changes in frequencies, clock pulses or voltages leading in and out of the chip[41].

The circuitry that handles the output of the tamper-response sensors is usually used to ensure that an attacker cannot obtain the secret data on the device. Often an attack is detected before the attacker has finished obtaining all the necessary data from the device, and in such cases it is essential that the device attempts to keep the attacker from obtaining the rest of the data. In most embedded devices and smart cards, the secrets are stored in either RAM or ROM memory modules. While RAM is relatively easy to clear during an attack, ROM is significantly harder.

The simplest way to erase the secrets in RAM is to do a *RAM Power Drop*. This means that power to the RAM modules is removed which effectively clears the contents.

A slightly more difficult way to clear RAM (or ROM) is by doing a *RAM Overwrite* (or *ROM Overwrite* respectively). A RAM overwrite repeatedly overwrites the memory module with all zeros and all ones alternatively. This process ensures that there is no residual information left that could be caused by imprinting, but it requires power and time to do the actual overwriting. This method is most accepted by governmental standards, but its success cannot be guaranteed in attack scenarios as a reliable source of power is needed while it is overwriting the memory modules.

The third and most effective way of ensuring that an attacker does not obtain the secrets on the device is by completely destroying the device itself it an attack is detected. Physical destruction of the device can be done by shorting certain parts of the circuit and thus rendering the device inoperable. It can be done with little to no violence, and in some cases may not even be detectable until the attacker notices that the device ceased functioning.

5.2.3 Tamper evidence

Tamper evident systems are designed to ensure that if a break-in occurs that evidence of the break-in is left behind. These systems do not protect against the attack itself, but only prove that an attack occurred after the fact. Tamper evident systems often use chemical or mechanical means to show evidence that an attack has taken place. As tamper evident systems themselves do not activate an alarm or otherwise notify the owner that a break-in attempt has occurred, it is important for an effective audit policy to be established and adhered to that visually checks the device frequently to ensure that there is no evidence of an attack[57]. As such tamper evident systems are often combined with a tamper response system to alert the owner of an attack, and to prove that an attack indeed took place.

As with the tamper resistance techniques there are a large number of different possibilities to ensure that tampering becomes evident. Again we will enumerate a number of possible methods. This list is incomplete as new materials are developed daily that can be used as a tamper evident layer. The use of cutting-edge materials can also help ensure that an attacker cannot easily replicate the material and reconstruct the tamper-evidence layer.

Brittle Packages The most trivial way of proving that a device has been tampered with is by sealing it in a brittle package. Once an attacker attempts to open or penetrate the enclosure the brittle package shatters and cannot be repaired. Such packages are difficult to reconstruct and thus the attacker leaves evidence of the attack.

Crazed Aluminum and Polished Packages The package is made from aluminum or other similar material, which has been heated (usually above 1000 degrees F.) and quenched. This heat treating causes a myriad of shallow, web-like cracks to appear on the surface. These cracks, like a fingerprint, are unique to each piece. The case can be photographed and subsequently audited using the photograph and optical comparison devices[57]. A polished package is an aluminum package that has been polished such that there are no cracks or marks evident. If on inspection there are such markings, it is evident that the package has been tampered with.

Bleeding Paint Paint of one color is mixed with micro-balloons containing paint of a contrasting color. If the painted surface is damaged by the attacker the other color bleeds onto the surface and is easy to identify as having been tampered with.

Holographic Tape The surface of tape, with a very firm adhesive, is printed with a holographic image similar to the kind used on credit cards. This kind of tape is moderately difficult to forge, and it is constructed so that attempts to remove it will damage it (the tape may be scored to promote tearing when removal is attempted)[57].

5.3 Countermeasures Against General Glitching

Countermeasures against fault injections are usually different from the countermeasures against side channel attacks. Below is a list of general countermeasures that are often used on smart cards and embedded devices.

- Low and high clock frequency sensor - Detects when the clock frequency leaves certain bounds, or if the frequency changes suddenly.
- High frequency signal detector - Detects high frequencies on buses and lines that should have relatively low frequency signals.
- Low and high temperature sensor - Detects fault injection attempts through temperature changes.
- Low and high supply voltage sensor - Detects attacks by monitoring for voltage changes in the power supply line.
- Light sensors (included integrated memory light sensor functionality) - Detect light that falls on the sensors or EEPROM memory cells to prevent Optical FI attacks.
- Electronic fuses for safeguarded mode control - Can turn off certain parts of the chip when triggered to prevent loss of sensitive information.
- Active shielding / Wire meshes - Detect probing attempts or attempts to decap the chip due to tiny wires being broken or short-circuited which is detected by the sensor. An example of a wire mesh is shown in Figure 5.3.
- Clock input filter for protection against spikes - Detects and prevents clock glitching attacks by ensuring that the external clock signal is “clean” and steady.
- Power input filter - Filters the power line’s input, to prevent spikes and high frequency pulses from entering the chip.
- Power-up and power-down reset - Resets the chip at every power-up and power-down to help prevent attacks in residual memory, or attacks that stack-up over multiple executions of the chip.

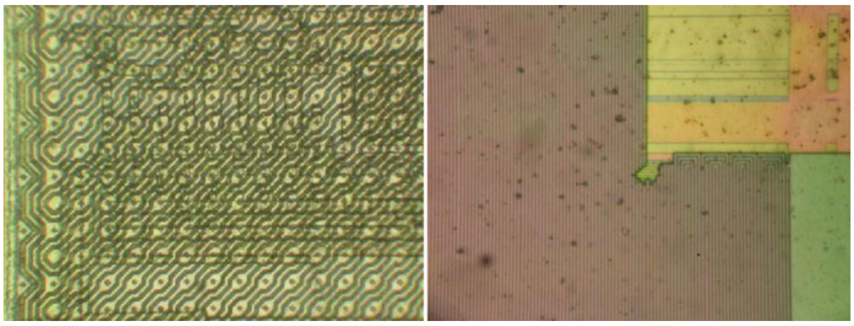


Figure 5.3: Top metal sensor meshes in smart card chips [54].

- Passivation detector - Detects attempts to decap the chip through chemical or abrasive means.
- Bus scramblers for EEPROM, RAM - Scramble the contents of RAM and EEPROM to prevent an attacker from understanding what is in memory.
- Memory Management and Protection Unit through MMU - Ensures that any changes in memory are not caused by an attacker by strictly managing the contents of memory with read/write access permissions.
- Epoxy encapsulation - Encapsulates the chip with a tough epoxy to prevent tampering and physical access to the chip surfaces.

5.4 Countermeasures Against Optical Fault Injection

Countermeasures against optical fault injection are usually based on blocking the line of sight to the chips' surface, or by detecting the optical fault injection through the influx of light onto light sensors. In the next sections we name two different techniques used to encapsulate the chip, but it should be noted that there are many such methods, and the two listed here are only given as examples of such techniques.

WORLD-RLC technology The RLC technology, shown in Figure 5.4, consists in a resin that is applied on top of the chip at the backside of the module after the connection of the chip to the backside of the contact plates has been completed. This resin is applied in a semi-liquid form. When applied it is covered with a glass epoxy patch and then polymerized by temperature. The resin and the patch are providing opacity that makes the silicon chip invisible when the module is finished. Its hardness provides efficient mechanical protection and tamper evidence; attempt to mechanically open the module will result in visible damage and/or loss of functionality by breaking of either silicon chip and/or wires. Before embedding into a smart card body, this module may be cut according two different shapes as shown in Figures 5.5a and 5.5b [6].

MIND-L Thermal black resin technology The thermal black resin technology, shown in Figure 5.6, consists in a resin that is applied on top of the chip at the backside of the module after the connection of the chip to the backside of the contact plates has been completed. This resin is applied in a semi-liquid form and is polymerized by temperature. This resin is characterized by its black color and opacity that makes the silicon chip invisible when the module is finished. Its hardness provides efficient mechanical protection and tamper evidence; attempt to mechanically open the module will result in visible damage and/or loss of functionality by breaking of either silicon chip and/or wires [6].

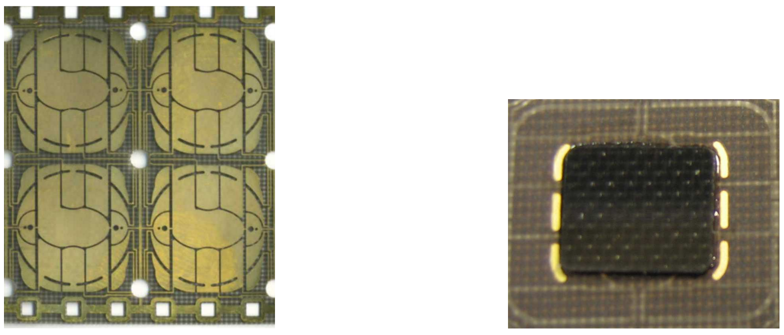
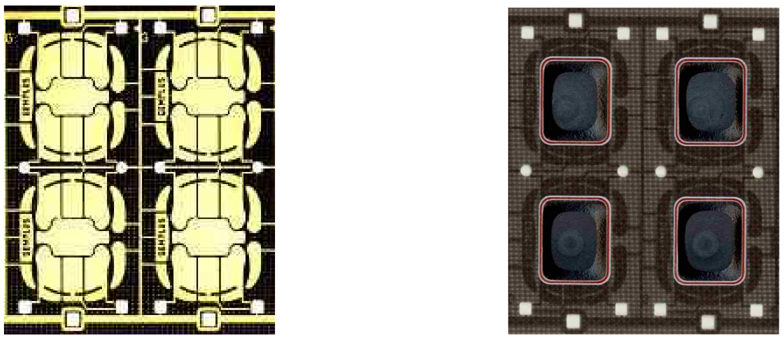

(a) Frontside (b) Backside

Figure 5.4: WORLD-RLC Technology. [6]

(a) "Oval"-cut variant. (b) "Square"-cut variant.

Figure 5.5: Variants of the WORLD-RLC Micropackaging. [6]



(a) Frontside (b) Backside

Figure 5.6: MIND-L Thermal Black Resin. [6]

Light sensors Embedded light sensors in the chip can detect photon emissions from lasers or other light sources and trigger security precautions on the device. These sensors usually cause a hard reset, but may also activate a security wipe procedure in which the RAM and EEPROM contents are wiped to ensure that no sensitive data is lost.

5.5 Countermeasures Against EM-FI

Reduction of the Electromagnetic Field Reduction of the electromagnetic field is relatively simple. It consists in using the metal levels that build the chip in order to reduce the radiation of the magnetic field. The metals usually used in the upper layers of processors are either aluminum or copper, which are not known for their ferromagnetic permeability. Their presence is enough to slightly reduce part of the electromagnetic field. [39] Note that the thickness of such materials can be limited by the maximum dimensions of the device, for example for smart card chips the thicknesses must be measured in nanometers as the entire chip modules is only around 0.2 mm.

Faraday Cage Imprisoning the chip in a Faraday cage is a sure method of keeping electromagnetic radiation from entering or leaving the chip, however the chip must be able to communicate with the environment, usually through a series of wired connections to the rest of the system. The construction of a Faraday cage around the processor concerns a non-commonplace exercise as it could not be perfect and must leave holes for the I/O wires. These holes allow for emanations of certain wavelengths to enter or escape the Faraday cage. Such a cage also requires extensive modifications during manufacturing [39].

Asynchronism Synchronous processes maximize the electromagnetic field which leaks from a chip. Similarly it is easier to cause glitches if you know when certain logic changes values, which is definitely the case if everything is synchronized to a clock pulse. This can be circumvented by using asynchronism. The processor no longer uses the internal clock to synchronize the operations and it is much more difficult for an observer to identify what is happening at what moment. Due to clock synchronization the power traces obtain a set of spikes, caused by all the processes continuing at the beginning of each clock cycle. Thanks to asynchronism these spikes dissolve because those processes no longer need to synchronize and thus there are far fewer spikes which can lead to information leakages. Without accurate timing EM-FI attacks are significantly more difficult as parameters such as clock cycles and offset no longer have any meaning.

Dual Line Logic Ensuring that for each line or bus that carries a signal there is a line in the opposite direction immediately next to it that carries the opposite signal is an effective way of lowering electromagnetic emissions as well as protecting against electromagnetic attacks. The two lines' electromagnetic fields interact to cancel each other out, practically removing most of the emissions. When an attacker tries to use EM-FI on such a pair of lines they will send the same pulse down both lines, effectively causing one line to spike to high values while canceling out the effect of the other line. A simple absolute comparator between the two line can then detect that there is a significant potential difference and signal an attack in progress.

5.5.1 Countermeasures Against Transient Pulse EM-FI

Typically countermeasures that work against VCC Glitching, Clock Glitching and General EM-FI help against the transient pulse variety of EM-FI. However these countermeasures need to be localized as the EM pulse can target only specific areas of the chip, instead of only on the wires that lead in and out of the device, and can thus evade the typical existing countermeasures.

5.5.2 Countermeasures Against Harmonic Emission EM-FI

For Harmonic emission EM-FI most of the countermeasures for Transient Pulse EM-FI and Harmonic VCC Glitching also have an effect. Additionally frequency sensors and additional filters could be placed in areas that may be targeted for attacks. Due to the wireless nature of EM-FI it is possible for an attacker to attack almost any wire on the chip, usually targeting the longest wires. By adding additional sensing and filtering components Harmonic EM-FI attacks may be mitigated.

5.6 Countermeasures implemented on Smart cards

To give an overview of what kind of countermeasures are typically found on smart cards, the next section shows the countermeasures on the different types of cards that were used during the experimentation in Chapter 6. Most smart card manufacturers try to keep the exact nature of the countermeasures secret to make it more difficult for an attacker to successfully attack the chip. Some card manufacturers do not list anything more than “This card has countermeasures against side channel and fault injection attacks” while other manufacturers list an extensive list of the types of sensors on the chip. None, however, post exact numbers of how many sensors are in use, or where they are located, etcetera.

5.6.1 Type A card

The Type A card does not have any countermeasures.

5.6.2 Types B, C, and E cards

The cards labeled B, C, and E are all cards for which the specific countermeasures are unknown. Their documentation states that the cards do have countermeasures, but it does not state what kind of countermeasures or what they protect against [6].

5.6.3 Type D card

The Type D card does have pretty extensive countermeasures. These countermeasures include a hard opaque epoxy protective material surrounding the chip, a cryptographic module that provides tamper resistance and tamper evidence, as required by FIPS 140-2 Level 3 physical validation, various hardware security mechanisms such as a clock input filter for protection against spikes and voltage, frequency, and temperature sensors, each with a high and low safeguard. Additionally it has SPA and DPA mitigation through random wait states on power up, write operations and power down resets. Timing attack mitigation through internal hardware timing to ensure every operation takes the same amount of time, and protects against fault injections through memory protection for RAM, EEPROM and ROM and signature techniques [6].

5.6.4 Type F and Type G cards

The Type F and Type G cards are based around similar chips [6]. Slight variants chips are common in the smart card industry, and their security features are pretty well defined.

- Enhanced security sensors such as low and high clock frequency sensors, low and high temperature sensors, low and high supply voltage sensors, light sensors (including integrated memory light sensor functionality), and Single Fault Injection attack detection sensors.
- Electronic fuses (for safeguarded mode control)
- Active shielding
- Unique die IDs
- Clock input filters (for protection against spikes)
- Power-up and power-down reset
- Programmable card disabling features

- Memory security for RAM, EEPROM and ROM (Both physical measures and encryption)
- Memory Management and Protection Unit
- Two different operation modes: System mode and User mode
- OS-controlled access restriction mechanism to peripherals in User mode
- Optional disabling of ROM read instructions by code executed in EEPROM
- Optional disabling of any code execution out of RAM
- No external clock for EEPROM
- Hardware sequencer controlled for EEPROM
- On-chip high voltage generation for EEPROM
- Enhanced error correction mechanism for EEPROM
- 14 B user write-protected security area in EEPROM (byte access, inhibit functionality per byte)
- 32 B write-once security area in EEPROM (bit access)
- 32 B user read-only area in EEPROM (byte access)

5.6.5 Type H card

The Type H card has a relatively vague description of the available countermeasures. They are "Probing detection" and "low frequency and supply voltage monitoring" [6]. The chip reacts to a low/high clock frequency, and low/high power supply voltage by resetting the cryptographic module. The specifications do not state what kind of probing detection is employed, nor whether the probing detection is only for physical access or if it also protects against electromagnetic probing. Similarly the frequency and voltage monitoring do not define where or how this monitoring takes place, meaning it could be only on the clock and power lines, but could also be spread randomly throughout the entire card.

5.6.6 Type I card

The Type I card has a pretty extensive overview of the built in security features [6].

- BlackResin Technology
- Shields - Active shield
- Hardware and enhanced security sensors
  - Passivation detector
  - Low/high voltage sensor
  - Low/high clock frequency sensor
  - Low/high temperature sensor
  - Light sensor
  - Glitch sensor
- Bus scramblers for EEPROM, RAM
- Memory Management and Protection Unit

A number of standard precautions are also built in to the chips logic.

- The chip provides the cryptographic module with hardware security mechanisms and reacts to those by resetting the cryptographic module. Any unprotected sensitive data is lost.
- The external clock frequency is monitored. If it is higher than the maximum value or lower than the minimum value, a security flag is raised.
- The supply voltage is monitored. If it is higher than the maximum value or lower than the minimum value, a security flag is raised.
- The temperature is monitored. If it is higher than the maximum value or lower than the minimum value, a security flag is raised.
- The active shield covers the ROM, the EEPROM and the analog blocks such as voltage regulator, oscillator and sensors.
- Light sensor is in the analogic part. The light sensor is hidden by the top metal layers of the circuit and cannot be distinguished by simple observation.
- Glitch sensor is present and monitors V_{CC} and V_{SS}. When the sensor is triggered a flag is raised.
- Filter is present on the RST (reset signal) and CLK (clock signal) lines.
- Logical addresses have no correlation thanks to the use of address scrambling at the BUS level.

5.6.7 Type J card

The Type J card has the following security features [6]:

- High frequency detector
- High voltage detector
- Low frequency detector
- High temperature detector
- Low temperature detector
- Epoxy encapsulation

The chip is embedded in epoxy, which completely encapsulates the whole Integrated Circuit (IC). Only micro-wires connecting to the faceplate penetrate the epoxy, connecting to the faceplate interface of the module. Attempts to tamper with the module result in damage to the epoxy, the plastic card, or the metal faceplate (scratches, chips, dents, etc.).

If the module is attacked through physical means, the attack will be evident due to the disturbance of the packaging of the card and module. The chip is embedded within an epoxy coating that is extremely difficult to penetrate without leaving evidence of the attack. Further, the packaging itself is resistant to penetration. Strong enclosure can be achieved by coating module components in a hard opaque tamper-evident epoxy. Physical removal of the epoxy will cause serious damage to the chip such that all cryptographic service providers are destroyed. Alternatively strong enclosure can also be achieved by a opaque tamper-evident covering of the die. Physical removal or tampering with the top metal layer will cause serious damage to the cryptographic module such that all cryptographic service providers are destroyed.

Chapter 6

Electromagnetic Fault Injection

6.1 Setup

In order to do an electromagnetic fault injection all that is really necessary is an electronic device and a controllable electromagnet. However, to produce measurable and repeatable results, a more extensive setup is required.

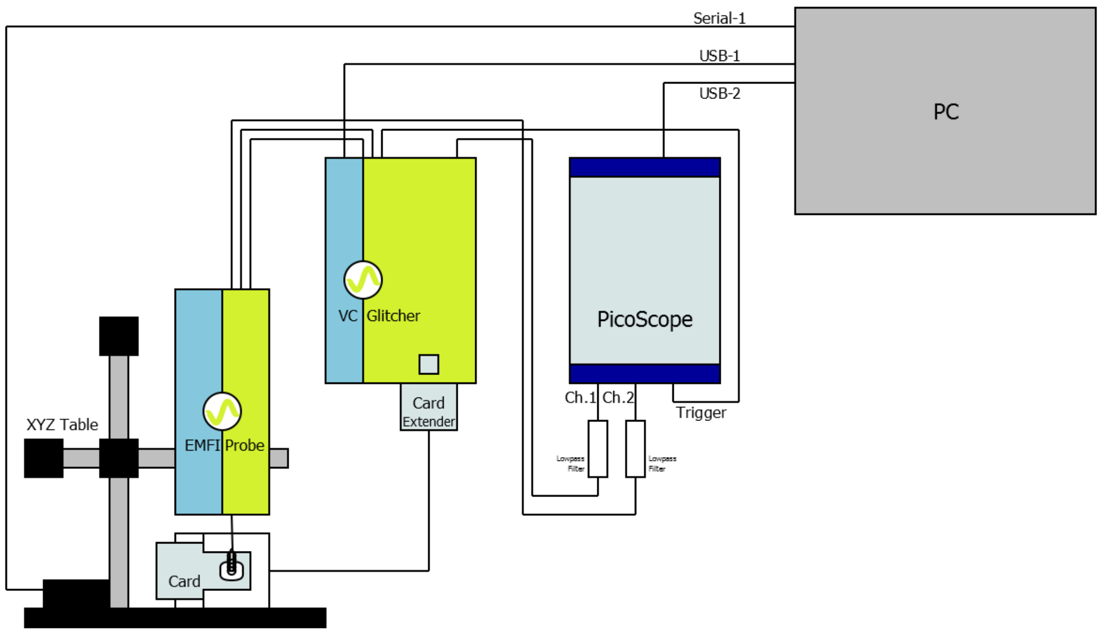


Figure 6.1: A standard configuration for fault injection on smart cards using the VC Glitcher, a Picoscope, a EM-FI probe, a XYZ-Table, and a set of low pass filters.

6.1.1 Hardware

For our research we used the setup shown in Figure 6.1 which consisted of the following devices and their respective USB and SMB-BNC cables:

- **Picoscope** - The Picoscope 5203 is a high performance 8-bit digital oscilloscope, capable of measuring 2 separate analog channels at 1 GS/s[1] [37].
- **VC Glitcher** - The VC Glitcher is a smart card reader made by Riscure B.V which is suited for all kinds of SCA en FI procedures. It is capable of generating accurate and repeatable 2 nanosecond voltage or clock glitches with fully programmable pattern and control logic [49].
- **XYZ Table** - The motorized XYZ table allows the probe to be positioned above the target with minute precision. The XYZ table has a precision of 2.5μm in all three dimensions, and has a repetition error of less than 50μm, to allow repeatability [44].
- **Smart card Extension Board** - The smart card extension board is simply a printed circuit board with pads that fits into a smart card reader in the same way a smart card would. The extension board has has a cable that connects it to an open smart card holder. This allows better access to the smart card under test without having to do the experiments within the immediate proximity of the actual smart card reader itself.
- **Electromagnetic Fault Injection Transient Probe** - The Electromagnetic Fault Injection Station is an electromagnetic probe that runs at 24VDC input voltage, which it transforms into a maximum of 450V and 64A output over the coil of the probe. It is capable of sending a 17 nanosecond EM pulse into a target with a frequency of 1 MHz. The pulse is capable of inducing voltages of up to -1.4V in the target device [50]. The EM-FI Probe has a number of interchangeable probe tips, with the coil in either a clockwise and counter-clockwise orientation. The tips used have a diameter of 1.5 mm and 4 mm.
- **Low Pass Filters** - We used a variety of inline BNC low pass filters to remove noise from the signals to be acquired.
  - Mini-Circuits 15542: BLP-50+ Low Pass filter, 50Ω, DC-48 MHz
  - Mini-Circuits 15542: BLP-1.9+ Low Pass filter, 50Ω, DC-1.9 MHz
  - ProbeMaster 1024: Termination, $50\Omega + 2\%$, 2W, 500 MHz
- **Smart cards** - We used smart cards of 10 different types during our research. Due to the sensitive nature of smart card security, the brands and models of the smart cards have been anonymized to prevent potential misuse.[2] The cards' specifications are shown in Table 6.1.

  The following history should be noted:
  - The Type D card is the successor of the Type B card.
  - The Type F card is the successor of the Type D card.
  - The Type G card is the successor of the Type E card.
  - The Type I card is the successor of the Type H card.

[1] GS/s: Gigasamples per second.
[2] Contact Riscure B.V. for more details on the card brand and type specifics.

Type	**Language**[a] **/ OS**[b]	**Certifications**[c d]	**Interfaces**
Type A	C code	-	Contact
Type B	JC 2.1.1, GP 2.0.1	-	Contact
Type C	JC 2.1.1, GP 2.0.1	-	Contact, Magnetic Strip
Type D	JC 2.2.1, GP 2.1.1	CC EAL 4+, FIPS 140-2 Level 3	Contact
Type E	JC 2.2.1, GP 2.1.1	CC EAL 4+	Contact, Contactless
Type F	JC 2.2.2, GP 2.1.1	CC EAL 5+	Contact
Type G	JC 2.2.2, GP 2.1.1	CC EAL 5+	Contact, Contactless
Type H	JC 2.2.1, GP 2.1	CC EAL 5+, FIPS 140-2 Level 3	Contact
Type I	JC 2.2.2, GP 2.1	CC EAL 5+, FIPS 140-2 Level 3	Contact
Type J	JC 2.2.1, GP 2.1.1	CC EAL 5+, FIPS 140-2 Level 3	Contact

[a]JC: JavaCard version
[b]GP: GlobalPlatform version
[c]CC EAL: Common Criteria Evaluation Assurance Level
[d]FIPS: Federal Information Processing Standard

Table 6.1: An overview of the properties of the Type A-J smart cards.

6.1.2 Software

In addition to the hardware mentioned above, Some additional software was used as well:

- **Inspector** - Inspector SCA & FI is an advanced integrated tool developed by Riscure for side-channel analysis and fault injection. It is designed to meet the highest standards of security research teams, certification labs and businesses around the world, and it excels in time-efficient analysis and perturbation of evaluation targets with the latest attack techniques and methods. Inspector supports side channel analysis methods such as power, timing, radio frequency, and electromagnetic analysis, and perturbation attacks such as voltage glitching, clock glitching, and laser manipulation [47].
- **Eclipse JCOP Tools** - IBM JCOP Tools 3.0 (JCOP Tools) provide a set of development tools for the successful development, testing, and deployment of applications for any generic OpenPlatform JavaCard, with specific support for the IBM JCOP platform [8].

In order to be able to consistently test the smart cards, they should be loaded with a similar program to attempt to glitch. Two different applications were used for testing purposes. The Type A smart card used an application based on C code which did a double pin verification, see Listing 6.1. All the other cards used an application written in Java, see Listing 6.2, which consisted of a loop within a loop which did some miscellaneous calculations in repetition. The calculations are done in such a way that it is possible to detect whether or not the card correctly executed the algorithm based on the results that are returned.

```
/* PIN implementation with auth flag, double verification using better comparison
    routine and try counter */
void better_pin_double(){
    if(pin_ctr>0) {
        --pin_ctr;
        eeprom_write_byte(&ee_pin_ctr, pin_ctr);
        if(array_compare(pin, buffer+DATA, 4)==0){
            if(buffer[P1]==0x01){
                int r = rand();
                eeprom_write_block(&ee_seed,(uint8_t *)&r, sizeof(int));
                _delay_loop_1(r & 0x0F );
            }
            if(array_compare(pin, buffer+DATA, 4)!=0){
                //Fail, fault injected!
                auth=FALSE;
                respond_code(0x00,0x69,0x86);
            } else {
                //Authentication complete!
                auth=TRUE;
                pin_ctr++;
                respond_code(0x00,0x90,0x00);
            }
        } else {
            respond_code(0x00,0x69,pin_ctr);
            auth=FALSE;
        }
    } else {
        auth=FALSE;
        respond_code(0x00,0x69,pin_ctr);
    }
}
```

Listing 6.1: C code for a double pin verification.

```
private void countInRAM(APDU apdu) {
    byte[] buff = apdu.getBuffer();
    apdu.setIncomingAndReceive();
    //Get counters from apdu
    short outerMax = Util.getShort(buff, ISO7816.OFFSET_CDATA);
    short innerMax = Util.getShort(buff, (short)(ISO7816.OFFSET_CDATA+2));
    //Init RAM variables
    short outerLoopCounterRAM = 0;
    short innerLoopCounterRAM = 0;
    short computation = 0;
    //Counting loop
    for(outerLoopCounterRAM = 0; outerLoopCounterRAM < outerMax;
        outerLoopCounterRAM++){
        computation=(short)-30;
        for(innerLoopCounterRAM = 0; innerLoopCounterRAM < innerMax;
            innerLoopCounterRAM++){
            computation=(short)-90;
            computation=(short)(computation+20);
            computation=(short)(computation-3);
        }
        computation=(short)(computation+53);
    }
    //Send results back
    short le = apdu.setOutgoing();
    if (le != 6) { //4 bytes for the counters + 2 for the computation
        ISOException.throwIt((short) (ISO7816.SW_WRONG_LENGTH));
    } else {
        apdu.setOutgoingLength((short) 6);
        Util.setShort(buff, (short)0, outerLoopCounterRAM);
        Util.setShort(buff, (short)2, innerLoopCounterRAM);
        Util.setShort(buff, (short)4, computation);
        apdu.sendBytes((short)0, (short)6);
    }
}
```

Listing 6.2: Java code for a double loop counter.

6.2 Procedure

In order to perform a successful electromagnetic fault attack on a smart card the properties of the device must first be identified as far as possible. The simplest way to do this is by checking the Answer to Reset (ATR) the card sends, to identify the type of card being used and possibly to identify the protocol which it is using. Once we have identified the protocol we can then attempt to identify a certain point in the operations where we wish to introduce a fault. Many smart cards are user-programmable which makes it easier to attempt EM-FI attacks, as we can program them with our own code. If we can upload our own applications then we can write an application such as the one listed in Listing 6.2. We now know exactly what the application on the smart card is supposed to do, so if we attempt to cause a fault while the program is running, we can see if the fault caused any changes in the expected output. In order to actually inject a fault, we first need to figure out which settings and which location we need to use for the EM probe. All smart cards will be run with an external clock of 1 MHz, and will use the T=0 protocol where available, or the T=1 protocol in the cases where T=0 is not available. Note that most cards have their own internal clocks which they use for the calculations, so the 1MHz clock is only used for I/O communications in most cases.

6.2.1 Identification of Card Characteristics

The first step in identifying the best settings to use to attack a smart card, is obtaining a power trace and an IO trace from the card. We log the messages sent to and from the smart card, and compare this to a plot of the power and IO signals. This allows us to see what the card is doing at which point in time. Listing 6.3 contains such a log of messages, which correspond to the traces in Figure 6.2. We can now attempt to make a correlation between the bytes in the listing with the spikes in the I/O trace. In the I/O trace of Figure 6.2 we can see three main spikes in the signal which indicate which part of the protocol is being output, and therefor we can also see when the smart card is working on a calculation. For instance, in Figure 6.2, the long period between the second and third spikes is where the code from Listing 6.2 is executed. We can now identify an approximate timing range during which we can attempt to introduce a glitch.

```
ATR:
 <  3B F9 18 00 FF 81 31 FE 45 4A 43 45 4E 53 4F 52 45 44 58
Prefer T=1 protocol; exchange characters at high speed:
 >  FF 11 18 F6
 <  FF 11 18 F6
Select ThisIsGoodCode (741515600DC0DE) application:
 >  00 00 0D 00 A4 04 00 07 74 15 15 60 0D C0 DE 00 50 00 00 02
Status OK:
 <  90 00 92
DoubleLoopCounter(2,1000):
 >  00 40 0D 00 BE 00 00 04 00 02 03 E8 06 5F
Return 2, 1000, -20:
 <  00 00 08 00 02 03 E8 FF EC
Status OK:
 <  90 00 62
```

Listing 6.3: Bytes transmitted over the I/O bus in T=1 protocol.

6.2.2 Location of Sensitive Areas

Theoretically we could simply start an exhaustive search over the search space defined by the different combinations of all the parameters, but such a search is unfeasible. In order to reduce the number of unknown parameters, we will do a number of small tests to decrease the search space. A number of techniques to decrease the size of the search space are available, as shown by

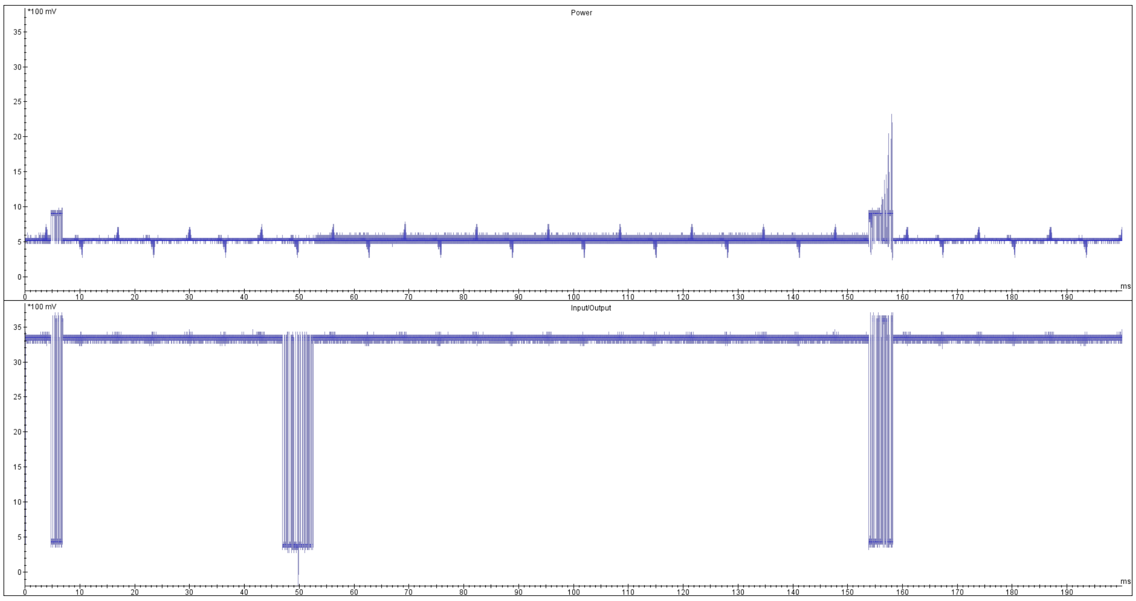


Figure 6.2: A trace from a single execution of the function in Listing 6.2 run on a Type G smart card. The top trace is the Power signal, the bottom trace is the I/O signal. A larger version of this figure is also included in the appendix as Figure D.7

Boix-Carpi in [9]. Our method was based on a trial and error process with manual interpretation of the results to optimize each parameter individually.

The next step in identifying the best settings for the parameters is to do a linear x,y-scan over the surface of the chip. Here the goal is to identify an area that is sensitive to electromagnetic fields. If the device under test has a large surface area, we can opt to use a large probe tip to find a sensitive area, and then switch to a smaller probe tip for more accurate positioning once we have a smaller range to scan. The choice of probe tips depends largely on the surface area of the chip, the chip's sensitivity and a number of other parameters, as shown by Carlier in [13].

For each location we execute the program on the card, and attempt to do a fault injection with a wide range of glitch parameters. Initially we start with a rough scan of say 5 by 5 or 10 by 10 points on the chip surface. In each point we can then step through a range of steps for a large number of parameters such as the glitch-power, the glitch-length, the wait-time, the glitch offset, and the number of glitches. We simply try a small set of parameters for which we make a plot of the results of the scan. We should be able to discern one or more areas that are more sensitive to electromagnetic radiation than the surrounding areas. Figure 6.4 displays such a plot [21], in which it is immediately obvious which locations are the most sensitive for the selected parameters. If we do not detect any sensitive areas with the current set of parameters we repeat the scan with different parameters until we find a setting that gives some kind of a result other than a normal output.

Once we have identified a promising location, we can then attempt to focus our search around the parameters that caused the successful faults during the x,y-scan. If we have a location that looks to be sensitive we can chose to focus on that one position, or we can do a secondary x,y-scan in that immediate area to try to find a better set of coordinates. If for some reason we fail to find any sensitive locations, even after multiple attempts, the EM-FI probe's power settings may be too low, or something may be wrong with the setup, or the smart card may have ceased operating. Almost all smart cards will show some sign of being glitched, either by giving a different response than the one we expected, or by doing a so-called *card mute* in which case the card simply stops responding to all input until the next reset. The latter usually results in a message timeout.

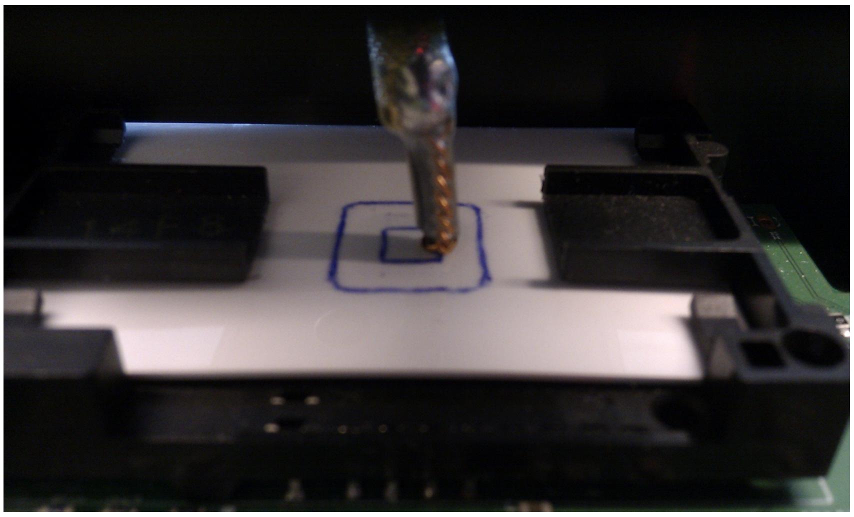


Figure 6.3: A photo of the EM-FI probe during a x,y-scan operation. The x,y-scan is used to detect sensitive areas on the device under test. The back of the smart card has the outline of the contact pads marked in pen for orientation.

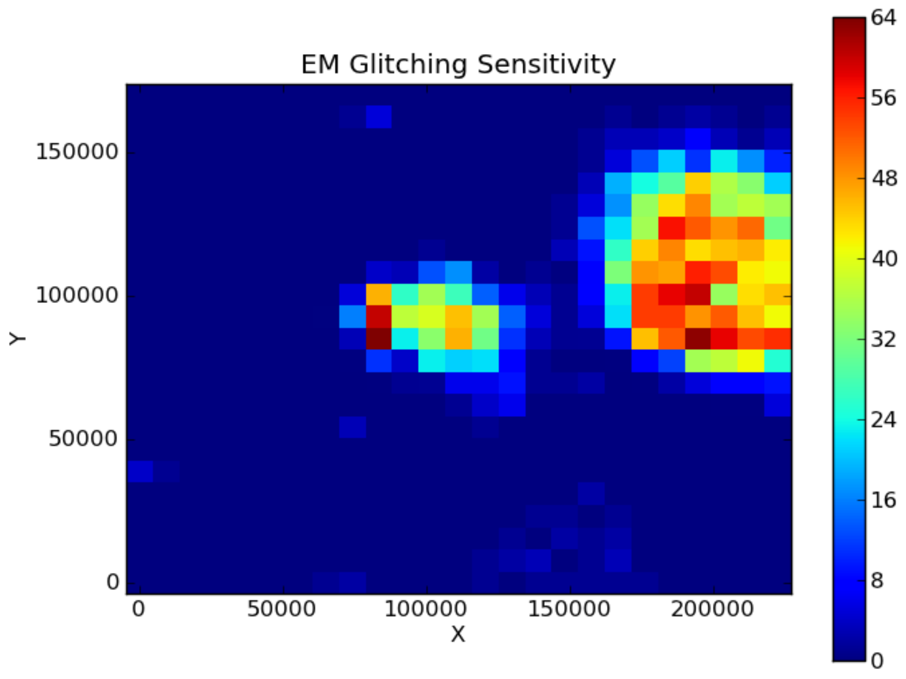


Figure 6.4: A plot of the sensitivity of a Type G card to an EM-FI probe during a x,y-scan operation. The coloring indicates the number of unexpected results that were acquired.

6.2.3 Selection of Parameters

The purpose of changing the parameters is to find a combination of settings such that the smart card gets affected by the EM-FI probes' pulse, but it does not cause a card mute event. A card mute event usually signals that the smart card has detected that it is operating outside of the standard operating specifications. A mute is a built-in safety procedure to prevent incorrect usage of the smart card, to prevent information leakage, and to keep any secret data on the smart card safe. Our goal is to find a combination of parameters so that we are on the border between those areas, as illustrated in Figure 6.5. Prior research at Riscure [9] shows that this boundary area usually gives the highest success rate. The figure illustrates a situation where only two parameters are being modified; a similar diagram can be made in 3D for 3 parameters, and in higher dimensions for more parameters. The boundary area, highlighted in yellow, is the area that we want to focus our attacks on as it usually leads to the most reliable results.

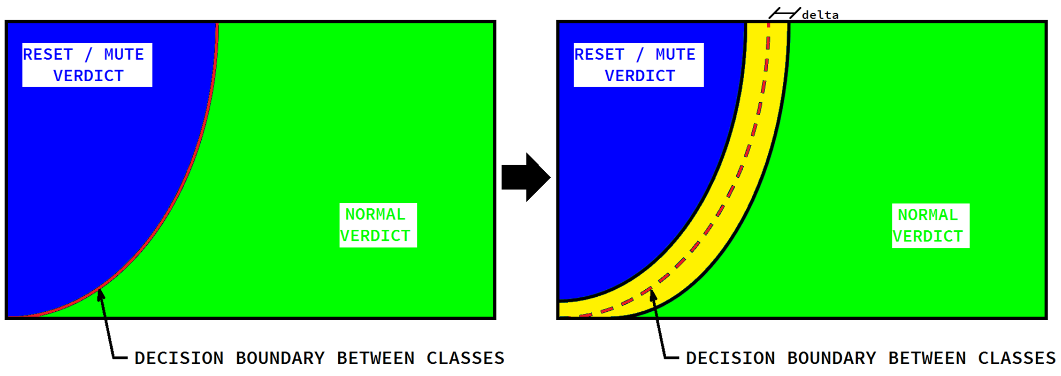


Figure 6.5: A diagram showing the responses for a range of two parameters on a smart card. [9] The left image shows the boundary area between normal responses and card mutes. The right image shows a yellow area which is interesting as it might contain parameters that can lead to a successful glitch.

6.2.4 Focused Attack

Once we have found a set of parameters that works to cause glitches that can be used, we can fine tune the parameters such as the timing and the number of glitches to modify the result of the glitches that are occurring. If we know exactly what the program is doing, we can use the resulting output to deduce which part of the program the glitch is affecting. By doing so we can identify the effect of a change in the parameters and steer toward an effect that can be used to modify the behavior of the smart card to suit our purposes.

6.3 Results

In the sections below the results for each of the different types of cards that we tested are listed. In most cases we did a number of different tests to get a good baseline and sample set for each card type. The results listed here are actual results from tests completed with the parameters shown, but they were not the only measurements and results gathered for each card type. These results should be read as "typical" results and the reader should note that even though there may be no further reference to it, additional testing of each card was indeed performed. In situations where the further testing led to interesting results, those results are described below as well.

6.3.1 Type A Card

Identification of Card Characteristics The Type A card is a C code based card with a contact interface only. This card has all countermeasures disabled and has several known vulnerabilities, including both VCC glitching and Optical FI. It is programmed with the function shown in Listing 6.1. The goal of the attack is to bypass the checks so that any pincode entered is accepted as a valid pincode. The attack must change the outcome of the boolean expressions on lines 6 and 12 in order for the attack to be qualified as successful. In the first check, on line 6, the result of the *array_compare()* function should return a value of *true* as the if-statement uses positive logic. In the second case, on line 12, the *array_compare()* function should return *false*, as that if-statement uses negative logic. This means that we will need to do two separate different glitches, to change the boolean expressions to *true* and *false* respectively. We executed the function 20 times, and analyzed plots of both the power and I/O inputs of the card to get an idea of the timing and potential delay jitter in the card. This led us to attempt glitches in the range of 700 to 900 wait cycles[3] after sending the command ADPU. This gives us a timing interval so that we can narrow down the search space, finding an exact timing is done later.

Location of Sensitive Areas & Selection of Parameters The Type A card was attacked in 4 different physical configurations. The initial tests were done using an unmodified Type A smart card, with attacks from both the front of the chip as well as attacks from the back of the chip. After initial testing, the tests were repeated with a decapped Type A smart card. The decapped Type A card is physically identical to the normal Type A cards except that there is a 4-by-4 millimeter square of material excised from the front of the card such that the backside of the chip itself is uncovered. This area is centered over the middle of the contact pads approximately matching the square area between contacts C2,C3 and C6,C7 in Figure 3.1, or the highlighted square in Figures 6.6c and 6.6d.

To find a suitable set of parameters, we used the following settings:

- x,y-scan: 10x10 measurements
- EM-FI Probe power settings: Range, 0% to 100% in steps of 1%.
- VCC Voltage: Range, 3V to 5V in steps of 0.5V
- Clock Voltage High: Linked to VCC voltage
- Clock Voltage Low: Fixed, 0V
- Wait cycles[3]: Random, number between 700 and 900
- Glitch cycles: Random, 2 to 10 pulses

[3]Reminder: The wait cycles are the clock cycles that pass before the EM pulse is triggered. The glitch offset is the number of nanoseconds into a certain clock cycle that should be waited before triggering the EM pulse. [i.e. If a pulse is triggered after 20 wait cycles, with a glitch offset of 30 ns, then the EM Probe will pulse 30 nanoseconds into the 21st clock cycle.] See Figure 4.1 for more details.

- Glitch offset[3]: Random, 0 to 500 nanoseconds
- Glitch length: Random, 20 to 30 nanoseconds
- Repetitions: 2

Each measurement was done twice for reliability. This led to a total of 101,000 measurements per side of each card, for a grand total of 404K measurements.

As described in Section 6.3.1 an x,y-scan was done on both the frontside and backside of both the normal and decapped smart cards. These scans led to the identification of one or two sensitive areas on each side of each card, as illustrated by Figure 6.6.

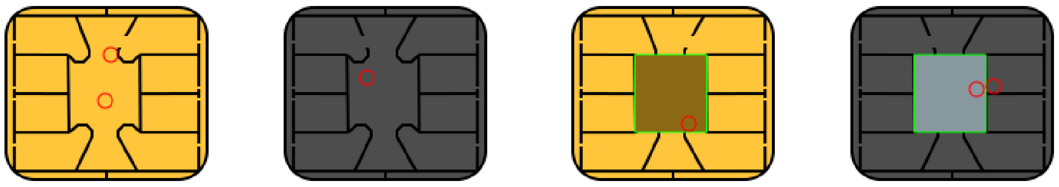

(a) Normal - Front (b) Normal - Back (c) Decapped - Front (d) Decapped - Back

Figure 6.6: Sensitive Areas for a Type A smart card.

Version	Power	First Glitch	Second Glitch	Settings
Normal Front	47%-85% VCC 3.0 V Clock 3.0 V	Wait Cycles: 788 Glitch cycles: 1	Wait cycles: 30-34 Glitch cycles: 4-6	Glitch offset: 70-80 ns Glitch length: 20-30 ns
Normal Back 3V	40%-100% VCC 3.0 V Clock 3.0 V	Wait Cycles: 788 Glitch cycles: 1	Wait cycles: 30-34 Glitch cycles: 4-6	Glitch offset: 70-80 ns Glitch length: 20-30 ns
Normal Back 5V	75% VCC 5.0 V Clock 5.0 V	Wait Cycles: 705-757 Glitch cycles: 1	Wait cycles: 5-35 Glitch cycles: 1	Glitch offset: 0-20 ns Glitch length: 20 ns
Decapped Front	45%-65% VCC 3.0 V Clock 3.0 V	Wait Cycles: 788 Glitch cycles: 1	Wait cycles: 30-34 Glitch cycles: 4-6	Glitch offset: 70-80 ns Glitch length: 20-30 ns
Decapped Back 3V	40%-100% VCC 3.0 V Clock 3.0 V	Wait Cycles: 788 Glitch cycles: 1	Wait cycles: 30-34 Glitch cycles: 4-6	Glitch offset: 70-80 ns Glitch length: 20-30 ns
Decapped Back 5V	65%-100% VCC 5.0 V Clock 5.0 V	Wait Cycles: 710-712 Glitch cycles: 1	Wait cycles: 15-25 Glitch cycles: 1-5	Glitch offset: 0-50 ns Glitch length: 20-40 ns

Table 6.2: Sensitivity settings for Type A smart card.

Initial testing also showed that the majority of the successful glitches occurred for the settings in Table 6.2. From these initial results we can see that the decapping of the chip does not have a really significant result for the Type A card. On the backside of the chip the decapping only helped to lower the power usage required by a mere 2% and on the frontside of the chip the effect of the decapping was negligible in these results.

It should be noted that the decapped card did allow for a greater overall number of glitches to take place for otherwise identical parameters. This was probably caused by the metal portion of the contact pads on the normal version of the card working as a shield for the electromagnetic radiation, thus the decapped card let more radiation actually reach the chip instead of dissipating most of the energy into eddy-currents in the contact pads.

While the decapped smart card did give somewhat better results, the decapping procedure is such a hassle that it does not weigh up against the benefits for EM-FI. We can also note that the attacks from the back of the card require less power than attacks focused on the front (contact pad) side of the smart card. This is probably caused by a combination of the distance between the probe tip and the active side of the chip, as well as the interference caused by the metallic pads on the frontside of the smart card.

Focused Attack Once we have decided on the bounds for the parameter ranges, and which configuration we want, we can then start doing a focused attack. For the Type A card we decided to use the normal version of the card, without decapping it, and to use the parameter ranges as found in Table 6.2. This led us to doing a 5-by-5 x,y-scan of the approximately 2-by-2 millimeter area around the position shown in Figure 6.6b with the parameters shown below:

- x,y-scan: 5x5 measurements, 2x2 mm area
- EM-FI Probe power settings: Range, 40% to 100% in steps of 1%.
- VCC Voltage: Fixed, 3V
- Clock Voltage High: Fixed, 3V
- Clock Voltage Low: Fixed, 0V
- Wait cycles Glitch 1: Fixed, 788
- Glitch cycles Glitch 1: Fixed, 1 pulse
- Wait cycles Glitch 2: Random, 30 to 34
- Glitch cycles Glitch 2: Random, 4 to 6 pulses
- Glitch offset: Random, 70 to 80 nanoseconds
- Glitch length: Random, 20 to 30 nanoseconds
- Repetitions: 5

The settings above result in a total of $5 * 5 * 61 * 5 = 7,625$ measurements. Of these $7,625$ traces 696 traces were classified as normal traces, where the EM pulse had no noticeable effect. $6,929$ traces were classified as interesting, where the pulse had some effect. Of those $6,929$ traces, $6,524$ resulted in errors, timeouts or other unexpected results. The remaining 405 measurements were considered successful glitches, as they caused the smart card to bypass both steps in the double pin verification. Figure 6.7 shows the voltage on the power and EM coil lines during a successful glitch. In that case we used a single glitch after 788 wait cycles, followed by a set of 5 pulses after an additional 32 wait cycles, while running the probe at 57% power on the backside of an undecapped smart card.

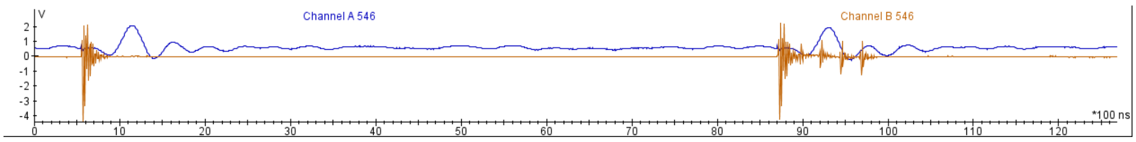


Figure 6.7: A Power and EM Coil Trace on a successful glitch of a Type A card. Channel A denotes the Power trace, Channel B is the voltage over the EM Coil. A larger version of this figure is also included in the appendix as Figure D.1.

Further testing of the Type A card also showed that in some cases it is possible to semi-permanently write a glitch to the card's memory. In some cases the conditional checks for the PIN were somehow written to the card, causing the card to accept any PIN code, until a new PIN was written to the card. This type of glitch was very rare and only occurred three times during the many tests we did on this card type. Additionally, one undecapped Type A card suffered from a permanently glitched Boot ROM, rendering it incapable of functioning normally. Possibly this card could have been recovered by flashing a new Boot ROM to the chip, but as it happened during testing we did not want to skew our results by using a refurbished card. All further testing was done using a new card.

Conclusions The results found during the focused attack lead us to believe that the Type A card is indeed vulnerable to successful fault injections using transient electromagnetic pulses. We are capable of introducing a number of different glitches in the execution of the code on the smart card in a reliable fashion, including changing the outcome of conditional statements using both positive and negative logic, both temporarily and semi-permanently, and thus the security and integrity of the output of the Type A smart card is no longer trustworthy.

6.3.2 Type B Card

Identification of Card Characteristics The Type B card is a JavaCard OS 2.1.1, GlobalPlatform 2.0.1 based card with a contact interface only. This card has unspecified countermeasures against fault injection and side channel attacks. It is programmed with the function shown in Listing 6.2.

The goal of the attack is to modify the output of the program so that the new output does not match the expected output. The attack can change the values of the variables *outerLoopCounterRAM*, *innerLoopCounterRAM*, or *computation*, or it can change the code flow itself by changing the results of the conditional statements and expressions. A third way the program flow can be modified is if the stack pointer itself is changed, causing the program to jump from one line of code to an arbitrary line instead of the next line. Often the latter type of glitch will result in the OS crashing as it is trying to execute arbitrary memory instead of java bytecode.

In order for the attack to be qualified as successful, we expect the program to give output in the expected format, but with one or more different numbers for the values of the variables. The function in Listing 6.2 requires a number of parameters to be passed to the function. We will consistently call the function with the parameters 2 and 1000, meaning that each trace actually consists of a repetition of 2 times the outer loop, which repeats the inner loop 1000 times each time. Each execution of the function will thus generate 6004 addition/subtraction operations in the loop, excluding the conditionals and loop counters.

We executed the function 20 times without attempting any attacks, and analyzed plots of both the power and I/O inputs of the card to get an idea of the timing and potential delay jitter in the card. Figure 6.8 shows the plots of four power and I/O traces. We can see that the card introduces a large amount of jitter between the sequential traces. The first spike in the figure represents the reply from the card to the select command, followed by the command ADPU (wide spikes) and the results to the command ADPU on the far right. If we synchronize our timings based on the select command, we have to deal with an average jitter of over 30 milliseconds; however, if we use the command ADPU to synchronize, we only have to deal with a minimal amount of jitter, and thus the timing of the attack can be specified to a much better degree. The difference can be seen if you compare Figures 6.8 and 6.9.

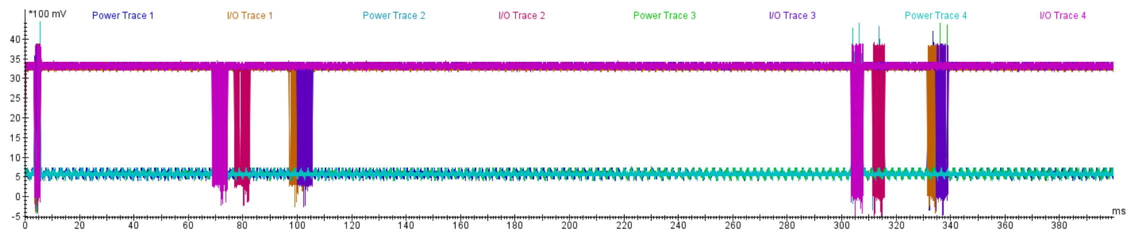


Figure 6.8: A set of 4 power and I/O traces for the Type B smart card, synchronized on the Select ADPU. This figure is also available full-page in the Appendix as Figure D.2.

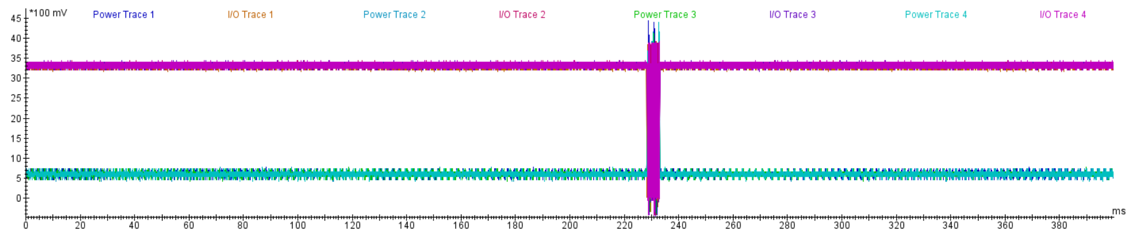


Figure 6.9: A set of 4 power and I/O traces for the Type B smart card, synchronized on the Command ADPU. This figure is also available full-page in the Appendix as Figure D.3.

The first is synchronized immediately after sending a select ADPU, in this plot the spikes for the command ADPU start around 65 ms and end around 105 ms, meaning that if we want to attack the processes during the double loop in the function, we must send the pulse after 105 ms to ensure that we are not glitching the card before the function has even started. Similarly, we cannot reliably attack the final iterations of the loops because we are not sure whether the function will complete after only 300 ms or after 340 ms. Due to those timing issues we effectively lose 40 ms at the beginning and 40 ms at the end of the function, while the entire loop only takes about 230 ms to complete. If we synchronize the start of the attack so that we use the end of the command ADPU to trigger the start of the timer, we obtain Figure 6.9. In this figure the spikes from Figure 6.8 are synchronized, and the plot starts immediately after the spikes, allowing us to utilize almost the entire 230 ms range for attacks.

We decided to use the variant that is synchronized immediately after the command ADPU, as this allows a much wider range of attack possibilities.

Location of Sensitive Areas & Selection of Parameters The Type B card was attacked from both the frontside and the backside of an unmodified smart card. The results from the Type A card and the Type F card (See section 6.3.6) led us to believe that the decapping procedure brings more risks than advantages and thus the tests were only done on an unmodified card.

To find a suitable set of parameters, we used the following settings:

- x,y-scan: 5x5 measurements
- EM-FI Probe power settings: Range, 20% to 100% in steps of 1%.
- VCC Voltage: Fixed, 5V
- Clock Voltage High: Linked to VCC voltage
- Clock Voltage Low: Fixed, 0V
- Wait cycles: Random, number between 1 and 20,000
- Glitch cycles: Random, 1 to 3 pulses
- Glitch offset: Random, 0 to 500 nanoseconds
- Glitch length: Fixed, 20 nanoseconds
- Repetitions: 2

Each measurement was done twice for reliability. This led to a total of $4,050$ measurements per side of the chip, for a grand total of $8,080$ measurements altogether.

The inital scans did not lead to any successful glitches. We did locate one area where the card was triggering more timeouts and card mutes than in the other areas so we decided to focus on that area, adjusting the parameters so that they were close to the boundary region between the normal traces and the muted traces as depicted in Figure 6.5.

Focused Attack As we were unable to find any parameters that led to a successful glitch, we decided to attempt a brute force scan over the boundary region between the normal traces and the muted traces. The brute force scan had the following parameters:

- x,y-scan: 3x3 measurements, 1.5x1.5 mm area
- EM-FI Probe power settings: Range, 35% to 48% in steps of 1%.
- VCC Voltage: Fixed, 5V
- Clock Voltage High: Fixed, 5V

- Clock Voltage Low: Fixed, 0V
- Wait cycles: Range, 1 to 200 in steps of 5
- Glitch cycles: Range, 1 to 3 pulses
- Glitch offset: Range, 0 to 500 nanoseconds in steps of 2 ns
- Glitch length: Fixed, 20 nanoseconds
- Repetitions: 0

The settings above result in a total of $3 * 3 * 14 * 50 * 251 = 6,852,300$ measurements. Of those $6,852,300$ traces there were zero traces that were classified as a successful glitch. All the traces were either a normal operations trace or a card mute in which the smart card stopped responding due to outside interference or internal errors.

This card was later tested again using almost identical parameters, and this led to the same results. Again no interesting traces or successful glitches were detected.

Conclusions The results found during the focused attack on the Type B card lead us to believe that it is not vulnerable to successful fault injections using this variety of transient electromagnetic pulses. The card seems to detect all our attempts to inject faults and thus we were unable to modify the outcome of the algorithm. We can conclude that the Type B card is relatively safe against EM-FI attacks.

6.3.3 Type C Card

Identification of Card Characteristics The Type C smart card is again a smart card that is based around JavaCard OS version 2.1.1 and GlobalPlatform 2.0.1. It features both a contact pad interface and a magnetic stripe. Our attack focuses on the contact pad interface, as the magnetic stripe does not interact with the embedded circuit on the smart card. This card also features a number of unspecified countermeasures against side channel attacks and fault injections and has the same application uploaded as is shown in Listing 6.2. The goal of the attack is the same as it was for the Type B card. We call the double loop function with 2 and 1000 recursions for the outer and inner loops respectively. The glitches should change the outcome of the calculation and thus the results returned by the application should differ from the expected results. Similarly to the Type B card, the Type C card also introduces randomized delays and jitter. Figures 6.10 and 6.11 show the differences in timing between the commands, as well as differences in processing time for individual pieces.

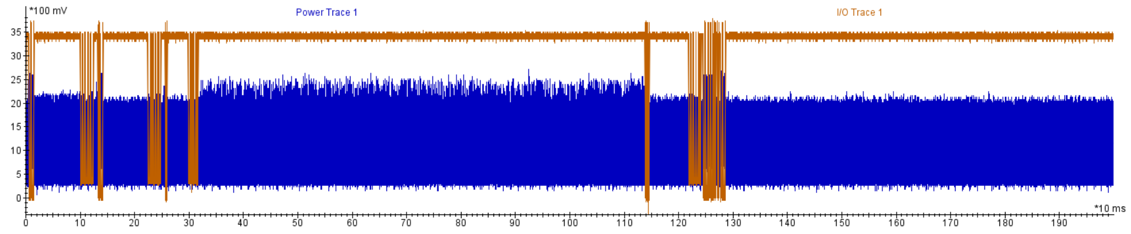


Figure 6.10: Power and I/O trace 1 for the Type C smart card, synchronized on the Select ADPU. This figure is also available full-page in the Appendix as Figure D.4.

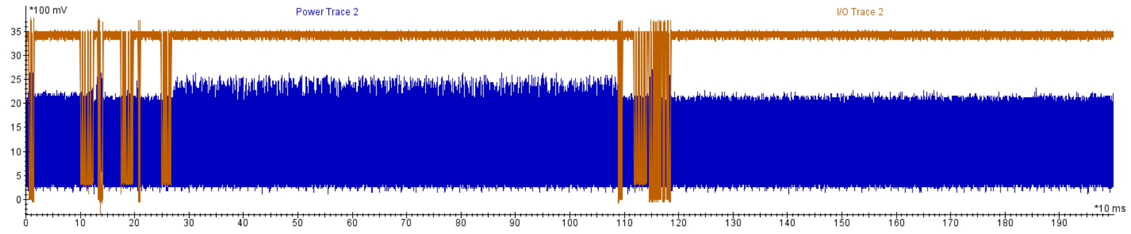


Figure 6.11: Power and I/O trace 2 for the Type C smart card, synchronized on the Select ADPU. This figure is also available full-page in the Appendix as Figure D.5.

Notice the differences between the three sets of orange spikes between 100 and 300 ms as well as the distances between the spikes near $1,100$ ms. Again synchronization on the command ADPU helps synchronize the majority of the trace, only the last bit of the trace where the result of the calculation gets returned has a bit of delay. Figure 6.12 shows that everything up to the crosshair near 830 ms is nicely synchronized, and only after the 830 ms mark do the individual traces start displaying their respective jitter. This means that we have a full 830 ms range to do our attack without having to worry about jitter interfering with the measurements.

Location of Sensitive Areas & Selection of Parameters The initial x,y-scan of the Type C card led to the conclusion that the chip was most sensitive in the center, on the back of the smart card, as Figure 6.13 indicates. It also appears to be the case that the metal pads on the frontside of the card are shielding a part of the magnetic radiation, as the interesting traces on the front of the card happen only at a much higher EM Coil power setting than the interesting traces on the backside of the card. The sensitive locations on the frontside of the card also appear near the edges of the metallic pads, leading us to believe that the metal surfaces are indeed hampering

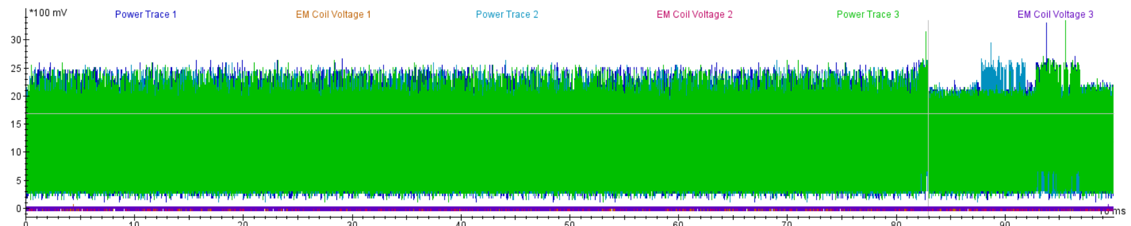
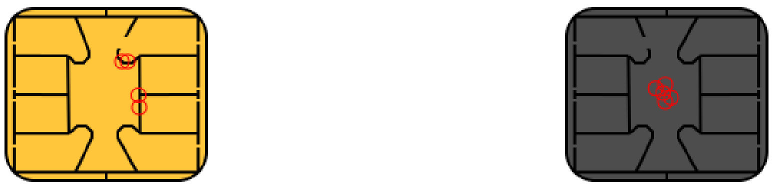


Figure 6.12: 3 Power and EM Coil Voltage traces for a Type C smart card, synchronized on the Command ADPU. This figure is also available full-page in the Appendix as Figure D.6.



(a) Front (b) Back

Figure 6.13: Sensitive Areas for a Type C smart card.

the EM waves. The difference in power between the frontside and the backside of the card varied with an average of 11% more power required on the frontside of the card.

Based on our initial x,y-scan during some basic testing we decided to use the following settings as the first set of parameters:

- x,y-scan: 10x10 measurements on back of card
- EM-FI Probe power settings: Range, 10% to 50% in steps of 1%.
- VCC Voltage: Fixed, 5V
- Clock Voltage High: Linked to VCC voltage
- Clock Voltage Low: Fixed, 0V
- Wait cycles: Random, number between 1 and 200
- Glitch cycles: Random, 1 to 10 pulses
- Glitch offset: Random, 0 to 50 nanoseconds
- Glitch length: Fixed, 20 nanoseconds
- Repetitions: 2

Focused Attack The x,y-scan gave a surprising number of interesting results, most noticeably on the back of the smart card. The scan itself already included a number of successful identical glitches, so we decided to focus an additional attack around those parameters to see if we could obtain any other unique successful results. We did an attack with the following settings:

- x,y-scan: Fixed, 1 point, centered above chip, back of card.
- EM-FI Probe power settings: Fixed, 25%.
- VCC Voltage: Fixed, 5V
- Clock Voltage High: Fixed, 5V
- Clock Voltage Low: Fixed, 0V

- Wait cycles: Range, 145 to 149 in steps of 1
- Glitch cycles: Range, 1 to 10 pulses
- Glitch offset: Range, 0 to 500 nanoseconds in steps of 2 ns
- Glitch length: Fixed, 20 nanoseconds
- Repetitions: 0

Conclusions The focused scan did a total of 12,050 measurements. Of those 12,050 measurements, 54 traces were classified as successful traces. These traces each resulted in the output ...`0x06 C0 00 02 03 03 FF EC 90 00` instead of ...`0x06 C0 00 02 03 E8 FF EC 90 00`, meaning that the outer loops did repeat both recursions, but the inner loop counter stopped repeating after 771 recursions instead of the usual 1000. This is most likely caused by glitching the condition $innerLoopCounterRAM < innerMax$ on line 14 of Listing 6.2 to evaluate to true before the expression is actually true. Even though we did not manage to cause any other types of glitches, we do know that certain types of glitches are possible, and the odds are good that with additional testing we would be able to find better parameters or other ways to break the system. The fact that we were capable of doing successful glitches remains, and thus we can conclude that the Type C card is vulnerable to EM-FI attacks.

6.3.4 Type D Card

Identification of Card Characteristics The Type D card is the successor of the Type B smart card. It is based on JavaCard OS version 2.2.1 with GlobalPlatform 2.1.1. It has a contact interface and has a Common Criteria Evaluation Assurance Level of 4+ and is certified FIPS 140-2 Level 3. It is loaded with the same Java application as cards B and C, with the code shown in Listing 6.2. We will use the standard 2 and 1000 repetitions for the outer and inner loops respectively. The type D card has an extensive list of countermeasures (see Section 5.6.3), internal delays and jitter. The latter two of which are largely evaded by triggering the EM pulses after synchronizing on the command ADPU.

Location of Sensitive Areas & Selection of Parameters In order to get an idea of which parameters we should use we did an initial x,y-scan over the backside of the smart card. Later that was followed up by an additional scan on the frontside of the smart card.

During our initial scan we used the following parameters:

- x,y-scan: 8x8 measurements on both the back and the front of the card
- EM-FI Probe power settings: Range, 0% to 100% in steps of 5%.
- VCC Voltage: Fixed, 5V
- Clock Voltage High: Linked to VCC voltage
- Clock Voltage Low: Fixed, 0V
- Wait cycles: Range, number between 0 and 2000 in steps of 5
- Glitch cycles: Random, 1 to 10 pulses
- Glitch offset: Random, 0 to 500 nanoseconds
- Glitch length: Fixed, 20 nanoseconds
- Repetitions: 2

Each measurement was done twice for reliability, leading to a total of $1,077,888$ traces on the backside of the smart card. The same scan was repeated on the front of the card, for a grand total of 2.145 million measurements. The initial scan on the backside of the card, and the follow-up scan on the frontside of the card did not give any interesting traces or successful glitches. All traces were either normal traces with the expected output, or the card muted or timed out to prevent misuse.

Focused Attack We have no parameters that lead to a successful glitch, so once again we decided to do a brute force attack above the center of the smart card's chip in the hopes of finding a set of parameters that cause a glitch which can be used in an attack. The brute force attempt used the following parameters:

- x,y-scan: 3x3 measurements on the back of the card
- EM-FI Probe power settings: Range, 0% to 100% in steps of 2%.
- VCC Voltage: Fixed, 5V
- Clock Voltage High: Linked to VCC voltage
- Clock Voltage Low: Fixed, 0V
- Wait cycles: Range, 300 and 500 in steps of 1
- Glitch cycles: Range, 1 to 10 pulses, steps of 2

- Glitch offset: Range, 0 to 500 nanoseconds steps of 25
- Glitch length: Fixed, 20 nanoseconds
- Repetitions: 0

The brute force scan consisted of 9,687,195 measurements. Once again the only results were normal traces and card mutes or timeouts, with zero interesting traces that denoted possible glitches in the execution of the application.

Conclusions From the results of both the initial scans and the brute force scan, we can draw the conclusion that the Type D card is not sensitive to electromagnetic fault injection using transient pulses. All attacks of sufficient power were detected using the built-in countermeasures.

6.3.5 Type E Card

Identification of Card Characteristics The Type E card is is based on JavaCard 2.2.1 and Global Platform 2.1.1. It has both a contact and a contactless interface, but we will only attack the contact interface. The Type E card has a Common Criteria Evaluation Assurance Level of 4+, and has multiple unspecified countermeasures against both fault attacks and side channel attacks. Once again the same Java application is loaded on the card, which we will execute using the same parameters as during earlier tests. Similarly to the Type B,C and D cards, this card has both delays and random jitter which makes exact triggering a bit more difficult. As we did earlier, we will trigger the attack after synchronizing on the command ADPU here as well.

Location of Sensitive Areas & Selection of Parameters Once again we did an x,y-scan over the both the backside and the frontside of the smart card to find a set of parameters we could use to narrow the scope of the attack.

The first few executions to determine the timing and jitter showed that the timing was almost identical to the Type B card. As such, we decided to use those parameters again:

- x,y-scan: 5x5 measurements
- EM-FI Probe power settings: Range, 20% to 100% in steps of 1%.
- VCC Voltage: Fixed, 5V
- Clock Voltage High: Linked to VCC voltage
- Clock Voltage Low: Fixed, 0V
- Wait cycles: Random, number between 1 and 20,000
- Glitch cycles: Random, 1 to 3 pulses
- Glitch offset: Random, 0 to 500 nanoseconds
- Glitch length: Fixed, 20 nanoseconds
- Repetitions: 2

Each measurement was done twice for reliability, for a total of $4,050$ measurements per side of the chip, for a grand total of $8,080$ measurements altogether.

Focused Attack Similarly to the Type B card, the scans did not lead to any successful glitches. Since we did not find any interesting measurements we decided to focus on the boundary areas using a brute force attack, to find out whether we missed the successful glitches due to having a grid that was too coarse, or parameters that weren't accurate enough. We decided to use the same parameters as for the Type B card:

- x,y-scan: 3x3 measurements, 1.5x1.5 mm area
- EM-FI Probe power settings: Range, 35% to 48% in steps of 1%.
- VCC Voltage: Fixed, 5V
- Clock Voltage High: Fixed, 5V
- Clock Voltage Low: Fixed, 0V
- Wait cycles: Range, 1 to 200 in steps of 5
- Glitch cycles: Range, 1 to 3 pulses
- Glitch offset: Range, 0 to 500 nanoseconds in steps of 2 ns

- Glitch length: Fixed, 20 nanoseconds
- Repetitions: 0

As was the case with the Type B card, of the 6, 852, 300 traces in the brute force scan, there were no traces that were classified as a successful glitch. All the traces were either a normal operations trace or a card mute in which the smart card stopped responding.

Conclusions The results found during the focused attack on the Type E card once again lead us to believe that it is not vulnerable to successful fault injections using this variety of transient electromagnetic pulses. The card seems to detect all our attempts to inject faults and all our glitches were unsuccessful. We can conclude that the Type E card is also relatively safe against EM-FI attacks.

6.3.6 Type F Card

Identification of Card Characteristics The Type F card is a 'current generation' JavaCard OS 2.2.2, GlobalPlatform 2.1.1 based JavaCard smart card with a Common Criteria Evaluation Assurance Level of 5+. The Type F card is the successor of the Type D card, which in turn was a successor of the Type B card. The Type F card is a contact only card and it has a number of different countermeasures against side channel and fault attacks, which include:

- Low and high clock frequency sensor
- Low and high temperature sensor
- Low and high supply voltage sensor
- Single Fault Injection (SFI) attack detection
- Light sensors (included integrated memory light sensor functionality)
- Electronic fuses for safeguarded mode control
- Active shielding
- Clock input filter for protection against spikes

The Type F card is loaded with the application in Listing 6.2 and is executed with respectively 2 and 1000 recursions for the outer and inner loops. After running the application on the card a number of times and looking at the power and I/O traces we decided to attack in the 2000 to 2500 wait cycles range.

Location of Sensitive Areas & Selection of Parameters To locate a sensitive area on the card, we scanned both sides of an unmodified card using an x,y-scan of 10x10 on a 5x5 mm area over the center of the chip. We also attempted to decap a number of Type F cards, but out of the 8 cards that we tried to use, 6 cards were destroyed during the decapping procedure and 2 cards were destroyed during testing with an Optical-FI setup. Due to the risks of destroying yet another card we decided to continue our EM-FI tests using only unmodified cards as they tend to survive significantly longer.

Initial scanning of the Type F card was done using the following parameters:

- x,y-scan: 10x10 measurements over a 5x5 mm area
- EM-FI Probe power settings: Range, 20% to 100% in steps of 5%.
- VCC Voltage: Fixed, 5V
- Clock Voltage High: Linked to VCC voltage
- Clock Voltage Low: Fixed, 0V
- Wait cycles: Random, number between 2000 and 2500
- Glitch cycles: Random, 1 to 10 pulses
- Glitch offset: Random, 0 to 500 nanoseconds
- Glitch length: Fixed, 20 nanoseconds
- Repetitions: 5

This led to an initial scan of $17,000$ traces, which included 7 successful glitches on the frontside of the chip, 1 successful glitch on the backside of the chip and a relatively high percentage of 21% of the traces resulting in card mutes or timeouts.

Focused Attack The focused attack was aimed at the parameters that gave successful glitches during the initial attack. A first focused attack was done using the following parameters:

- x,y-scan: Fixed, Sensitive coordinates found during initial scan
- EM-FI Probe power settings: Fixed, 75%.
- VCC Voltage: Fixed, 5V
- Clock Voltage High: Linked to VCC voltage
- Clock Voltage Low: Fixed, 0V
- Wait cycles: Range, number between 2005 and 2025 in steps of 1
- Glitch cycles: Fixed, 5 pulses
- Glitch offset: Range, 0 to 500 nanoseconds in steps of 2 ns
- Glitch length: Fixed, 20 nanoseconds
- Repetitions: 10

Of these 52, 710 traces, 4 were successful glitches. We further focused the attack to the parameters:

- x,y-scan: Fixed, Sensitive coordinates found during initial scan
- EM-FI Probe power settings: Fixed, 75%.
- VCC Voltage: Fixed, 5V
- Clock Voltage High: Linked to VCC voltage
- Clock Voltage Low: Fixed, 0V
- Wait cycles: Range, number between 2008 and 2015 in steps of 1
- Glitch cycles: Fixed, 5 pulses
- Glitch offset: Range, 0 to 1000 nanoseconds in steps of 2 ns
- Glitch length: Fixed, 20 nanoseconds
- Repetitions: 10

This resulted in 19 successful glitches, most of which were at 2011 wait cycles. During the inital scanning, we also found a successful glitch at 2340-2341 wait cycles. We decided to try a pinpoint scan, just to see how many successful glitches could be achieved by doing a very small but intensive scan on a set of parameters that showed promising. The parameters used were:

- x,y-scan: Fixed, Sensitive coordinates found during initial scan
- EM-FI Probe power settings: Fixed, 75%.
- VCC Voltage: Fixed, 5V
- Clock Voltage High: Linked to VCC voltage
- Clock Voltage Low: Fixed, 0V
- Wait cycles: Range, 2340 to 2341
- Glitch cycles: Range, 1 to 10 pulses
- Glitch offset: Range, 0 to 500 nanoseconds in steps of 2 ns
- Glitch length: Fixed, 20 nanoseconds
- Repetitions: 2

This pinpoint attack led to $1,004$ measurements being done, of which 13 were successful glitches.

The results also showed that the successful glitches can have different outcome results for otherwise identical parameters. A normal trace should give the result ... `0x40 08 00 02 03 E8 FF EC 90 00 22`. This means that the function in Listing 6.2 completed successfully, with the results 2, 1000, and -20 for the *outerLoopCounterRAM*, *innerLoopCounterRAM*, and *computation* variables respectively. However, the glitched traces reported the following results:

```
1  ...40 02 90 00 D2                      (Glitched entire function)
2  ...40 08 00 00 00 00 00 00 90 00 D8  (Glitched entire outer loop)
3  ...40 08 00 02 00 00 00 17 90 00 CD  (Glitched entire inner loop)
4  ...40 08 00 02 00 09 FF EC 90 00 C0  (Glitched inner loop after 9 loops)
5  ...40 08 00 02 03 12 FF EC 90 00 D8  (Glitched inner loop after 786 loops)
6  ...40 08 00 02 04 00 FF EC 90 00 CD  (Glitched inner loop after 1024 loops)
7  ...40 08 00 06 03 E8 FF EC 90 00 26  (Glitched outer loop after 6 loops)
8  ...40 08 01 04 00 01 FF EC 90 00 CF  (Glitched outer loop after 260 loops)
```

Listing 6.4: Successful glitch results on a Type F smart card

Conclusions From Listing 6.4 we can see that there are a number of different types of results. The result on line 1 is a glitch that causes the Java application to skip over the entire function, to just return a `0x90 00` response without doing the actual calculations or returning the results. This type of glitch would be catastrophic in applications that call a certain function to check for permissions or similar. The attacker could simply skip that call and proceed as if they have the permissions.

The glitch on line 2 causes the application to proceed through the loop structure immediately, as it thinks that the first loop condition on line 12 of Listing 6.2 is true. As a result the variables are not set to the expected values and only zeros are output by the application. The glitch result on line 3 does the same to the inner loop condition. It returns immediately instead of repeating 1000 times.

Result lines 4 to 6 all cause the inner loop to misinterpret the condition for repetition, causing 9, 768, and 1024 loops respectively. Note that in the latter case the loop does more repetitions than it was supposed to. This cannot be done by simply changing the value of the condition once, so in the last case the actual value of the *innerMax* variable must have been changed in memory. A similar situation occurs for result line 7, but in this case the outer loop is repeated 6 times. Again this must be a "permanent" change in memory, as the application would have terminated after the third recursion if the condition itself had been modified. As each recursion of the outer loop contains 1000 recursions of the inner loop, it is very unlikely that this glitch was not caused by a modified variable.

The last line of Listing 6.4 contains a glitch that caused the function to repeat the outer loop 260 times. The inner loop, which should do 1000 recursions, only completed a single recursion during the last recursion of the outer loop. As this can only happen if both the conditions for the first and second loops evaluate to true, this means that both the *innerMax* and *outerMax* variables were probably modified by the glitch.

We managed to produce a broad range of different glitch results with such a simple program, some of which can modify the code flow in an application on the smart card or cause the application to skip entire portions of the code. As this breaks both the data integrity and the security of the software itself we deem that the Type F smart card is severely sensitive to EM-FI and should avoid being used.

Additionally, as the Type F card has a list of known countermeasures, it shows that EM-FI is capable of subverting those countermeasures and can still produce successful attacks even with extensive countermeasures in place. Testing of this same card using regular Voltage and Clock glitching did not give any successful results, proving that the countermeasures did in fact work against VCC glitching. Similarly attempts to attack the Type F card using optical fault injection methods failed, due to the incredible difficulty of decapping a Type F smart card and the countermeasures that were triggered by the laser used during the tests.

6.3.7 Type G Card

Identification of Card Characteristics The Type G card is also a 'current generation' JavaCard OS 2.2.2, GlobalPlatform 2.1.1 based JavaCard smart card with a Common Criteria Evaluation Assurance Level of 5+. This dual interface (contact & contactless) card is the successor of the Type E card, and is closely related to the Type F card. The Type G card the same countermeasures against side channel and fault attacks as the Type F card. The Type G card is also loaded with the application in Listing 6.2 and is executed with respectively 2 and 1000 recursions for the outer and inner loops. After running the application on the card approximately 20 times and looking at the power and I/O traces we decided to attack in the 95, 000 to 100, 000 wait cycles range. Figure 6.14 shows a power and I/O trace of this card.

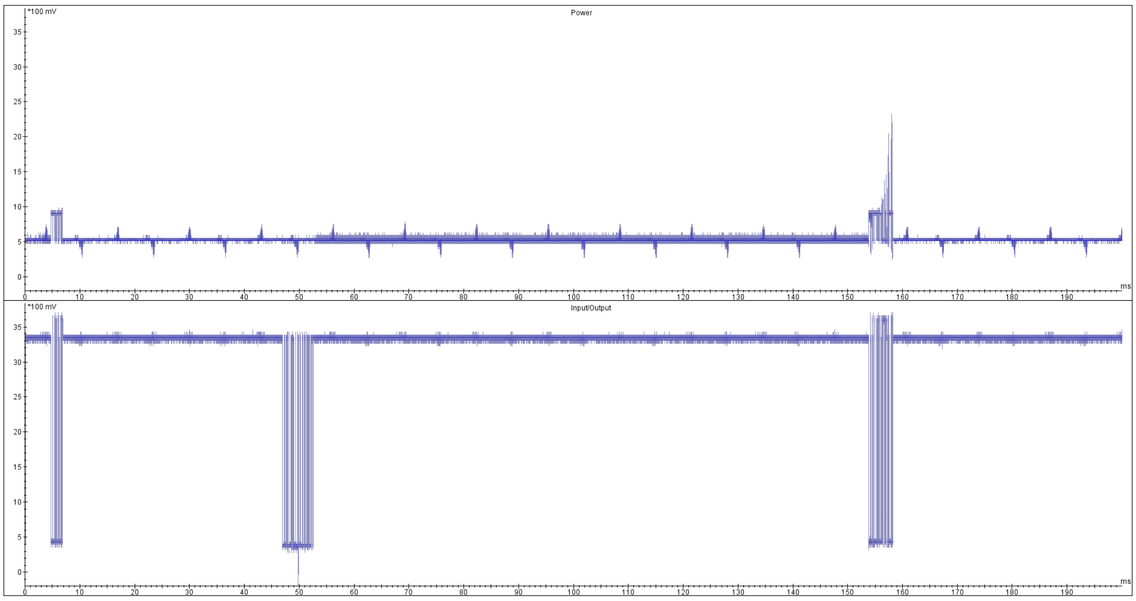


Figure 6.14: A Power and I/O trace from a single execution of the function in Listing 6.2 on a Type G smart card. This figure is also available full-page in the Appendix as Figure D.7.

Location of Sensitive Areas & Selection of Parameters As was the case with the Type F card, we decided to do an x,y-scan on both the front and the back of the card. Based on the fact that the Type F card was relatively sensitive, and this card is a sibling from the same generation, we attempted a scan of 5x5 first, to see if that would give us any results as a starting point.

Initial scanning of the Type G card was done using the following parameters:

- x,y-scan: 5x5 measurements over a 4x4 mm area
- EM-FI Probe power settings: Range, 35% to 85% in steps of 1%.
- VCC Voltage: Fixed, 5V
- Clock Voltage High: Linked to VCC voltage
- Clock Voltage Low: Fixed, 0V
- Wait cycles: Random, number between 95,000 and 100,000
- Glitch cycles: Random, 1 to 3 pulses
- Glitch offset: Random, 0 to 500 nanoseconds

- Glitch length: Fixed, 20 nanoseconds
- Repetitions: 5

This led to an initial scan of 25,500 traces, which included 2 successful glitches on the frontside of the chip and 0 successful glitches on the backside of the chip. Again we found a relatively high percentage of approximately 18% of traces which resulted in timeouts or card mutes.

Focused Attack The inital scan pointed us to two specific areas that looked promising. The first area gave the following set of parameters to focus on:

- x,y-scan: Fixed, Sensitive coordinates found during initial scan
- EM-FI Probe power settings: Fixed, 85%.
- VCC Voltage: Fixed, 5V
- Clock Voltage High: Linked to VCC voltage
- Clock Voltage Low: Fixed, 0V
- Wait cycles: Range, number between 95,000 and 98,000 in steps of 1
- Glitch cycles: Range, 1 to 3 pulses
- Glitch offset: Random, 0 to 500 nanoseconds
- Glitch length: Fixed, 20 nanoseconds
- Repetitions: 2

Those settings led to 18,000 traces, of which 2 were considered successful. We then tried a focused attack at the other location we found during the inital testing, using the following parameters:

- x,y-scan: Fixed, Sensitive coordinates found during initial scan
- EM-FI Probe power settings: Fixed, 75%.
- VCC Voltage: Fixed, 5V
- Clock Voltage High: Linked to VCC voltage
- Clock Voltage Low: Fixed, 0V
- Wait cycles: Range, number between 95,000 and 96,000 in steps of 1
- Glitch cycles: Fixed, 1 pulse
- Glitch offset: Range, 0 to 500 nanoseconds in steps of 2 ns
- Glitch length: Fixed, 20 nanoseconds
- Repetitions: 2

The second focused scan gave another 502,502 results, of which again 3 were classified as successful.

Conclusions As was the case with the Type F card, the Type G card is also protected with numerous countermeasures against fault injection attacks. It appears that these countermeasures are not sufficient to protect the smart card against EM-FI attacks.

The successful glitches found during the focused attacks were all of a similar genre, each only affected the termination of the inner loop, but the timing was accurate enough to define on which recursion the inner loop was supposed to terminate.

```
...40 08 00 02 03 56 FF EC 90 00 9C (Glitched inner loop after 854 loops)
...00 08 00 02 03 5A FF EC 90 00 D0 (Glitched inner loop after 858 loops)
...40 08 00 02 03 78 FF EC 90 00 B2 (Glitched inner loop after 888 loops)
...40 08 00 02 03 86 FF EC 90 00 4C (Glitched inner loop after 902 loops)
...40 08 00 02 03 87 FF EC 90 00 4D (Glitched inner loop after 903 loops)
```

Listing 6.5: Successful glitch results on a Type G smart card

Listing 6.5 shows that the results obtained have the inner loop terminating after respectively 854, 858, 888, 902, and 903 recursions. The glitches that caused the 854- and the 858 cycle loops were almost identical, both were caused by a single glitch after 95, 500 wait cycles. The only difference between the two was the offset value, in this case the difference was a mere 384 nanoseconds. Conversely the timing difference between the 902- and the 903 iteration loops was approximately 129 wait cycles, with 3 and 2 glitch cycles respectively, so some tinkering is required in order to find the exact timing necessary to break the loop at a precise location.

Once again we have found a number of successful glitches on a smart card with countermeasures against fault injections. As we can reliably reproduce these glitches, and can even select on which iteration we want to break out of a loop, we feel that the security and integrity of this smart card is no longer trustworthy. The Type G card should be avoided.

6.3.8 Type H Card

Identification of Card Characteristics The Type H smart card is based on JavaCard OS version 2.2.1 with GlobalPlatform 2.1. It has both a Common Criteria Evaluation Assurance Level of 5+ as well as a FIPS 140-2 Level 3 rating. It is a contact only card. The Type H card has the following countermeasures:

- Probing detection
- low frequency monitoring
- low supply voltage monitoring
- reacts to low/high clock frequency by resetting the cryptographic module
- reacts to low/high power supply voltage by resetting the cryptographic module.

The Type H card is loaded with the Java application in Listing 6.2 and is called using the same parameters as were used in all the other tests. An initial plot of the power and I/O traces of three executions of the application showed that the card output a huge amount of delays and jitter, so much that it was very difficult to discern when one operation ended and the next started, as can be seen in Figures 6.15 and 6.16. Synchronizing on the command ADPU helped, but there is still a significant amount of jitter present.

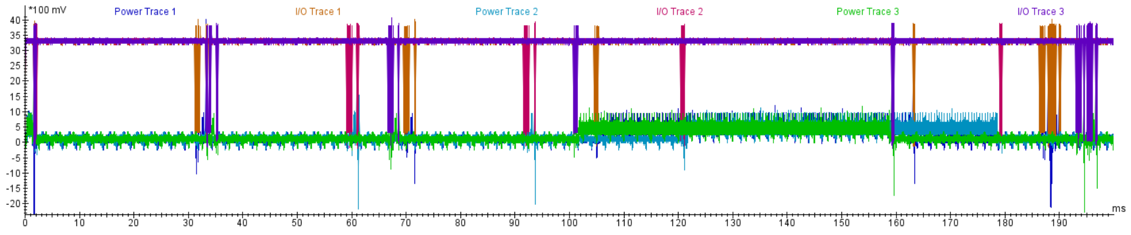


Figure 6.15: A set of 3 power and I/O traces for the Type H smart card, synchronized on the Select ADPU. This figure is also available full-page in the Appendix as Figure D.8.

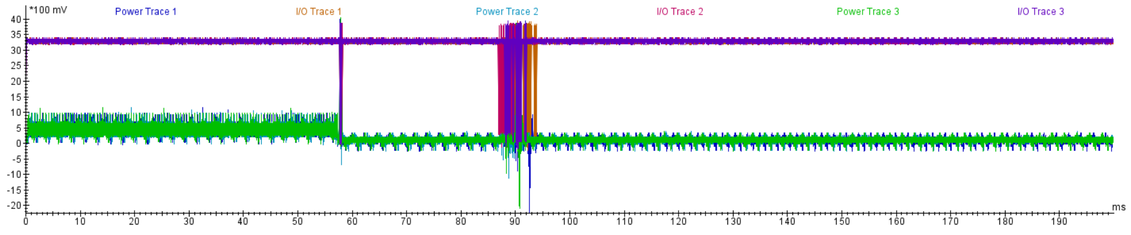


Figure 6.16: A set of 3 power and I/O traces for the Type H smart card, synchronized on the Command ADPU. This figure is also available full-page in the Appendix as Figure D.9.

As we were uncertain where to start our attack, we decided to use the entire range of 1 to 60,000 wait cycles for our initial scan, and then narrow down the search space once the first interesting traces had been found.

Location of Sensitive Areas & Selection of Parameters For the Type H smart card we decided to attempt a 5x5 x,y-scan, over a 4x4 mm area, on the frontside of the chip.

Initial scanning of the Type H card was done using the following parameters:

- x,y-scan: 5x5 measurements over a 4x4 mm area
- EM-FI Probe power settings: Range, 40% to 85% in steps of 5%.

- VCC Voltage: Fixed, 5V
- Clock Voltage High: Linked to VCC voltage
- Clock Voltage Low: Fixed, 0V
- Wait cycles: Range, 1 to 61,000 in steps of 23
- Glitch cycles: Fixed, 1 pulse
- Glitch offset: Random, 0 to 500 nanoseconds
- Glitch length: Fixed, 20 nanoseconds
- Repetitions: 1

This led to an initial scan of 663, 055 traces, which included only 1 successful glitch. We also found a large number of card mutes, all of which started at approximately 75% EM Coil power usage.

Focused Attack For the focused attack we decided to focus on the area that gave the successful glitch, and to do the entire initial scan again, but to focus on the boundary condition of 75% coil power. This led to the following set of parameters:

- x,y-scan: Fixed in area defined by initial scan.
- EM-FI Probe power settings: Range, 73% to 77% in steps of 1%.
- VCC Voltage: Fixed, 5V
- Clock Voltage High: Linked to VCC voltage
- Clock Voltage Low: Fixed, 0V
- Wait cycles: Range, 1 to 61,000 in steps of 1
- Glitch cycles: Fixed, 1 pulse
- Glitch offset: Random, 0 to 500 nanoseconds
- Glitch length: Fixed, 20 nanoseconds
- Repetitions: 1

The focused attack gave us a total of 305, 000 results, of which 5 were classified as successful glitches. Over 60% of the traces at 75% coil power led to card mutes or timeouts. At 76% coil power this was almost 98%, leading us to believe that we were triggering a voltage sensor or other similar countermeasure in the card.

Conclusions The focused attack gave us 5 successful glitches, of which there were 4 unique results and 1 was a repetition of a result obtained earlier. Listing 6.6 shows the results

```
...06 C0 00 02 00 E4 FF EC 90 00 (Glitched inner loop after 228 loops)
...06 C0 00 02 01 44 FF EC 90 00 (Glitched inner loop after 324 loops)
...06 C0 00 02 01 66 FF EC 90 00 (Glitched inner loop after 358 loops)
...06 C0 00 BE 00 00 04 00 90 00 (Glitched inner loop to repeat 190 times, and set
    'computation' to 1024 instead of -20)
```

Listing 6.6: Successful glitch results on a Type H smart card

The last result in the listing was found twice, leading us to believe that the loops were in fact modified. However, we believe that instead the card OS was glitched in such a way that it returned a part of the command ADPU string instead of the actual values calculated during the loop. The I/O trace contained `0xC0 00 00 00 6D 00 00 BE 00 00 04 BE 00 02 03 E8 61`, of which the `0x00 BE 00 00 04` is apparently echoed in place of the results we expected. If this is indeed the case, it means that we are capable of inserting arbitrary bytes into the response of the application. Even if this is not the case, we are capable of producing reliable glitches on this card, and thus we cannot ascertain the integrity and security provided by this card. We suggest that the Type H card be deprecated.

6.3.9 Type I Card

Identification of Card Characteristics The Type I smart card is based on JavaCard OS version 2.2.2 with GlobalPlatform version 2.1. It has a Common Criteria Evaluation Assurance Level of 5+ and is FIPS 140-2 Level 3 rated. This is a card with only a contact interface and is the successor of the Type H smart card. The version of the card that we tested was the SIM-cut variant, ID-000, in an ID-1 holder, denoted as ID-1/000, but this should not have any effect on the security of the card itself.

While the Type H card had a small set of countermeasures, the newer Type I card features an extensive suite of countermeasures as noted in Chapter 5.6.6. An initial trace did not really help to tell at what time we should attack the card. This card creates delays in such a way that even if we synchronize on the command ADPU we still do not have any clear indications of what is happening at which point in time, and so we will have to estimate when to attack. Figure 6.17 shows the delays between 4 consecutive traces when using synchronized command ADPUs.

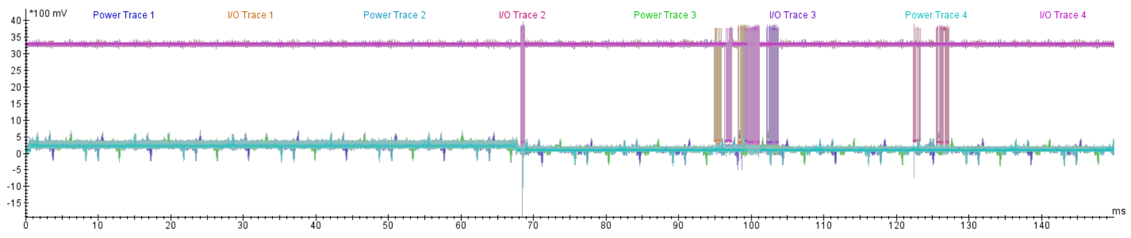


Figure 6.17: A set of 4 power and I/O traces for the Type I smart card, synchronized on the Command ADPU. This figure is also available full-page in the Appendix as Figure D.10.

Location of Sensitive Areas & Selection of Parameters We decided to do an x,y-scan of 5x5 over the frontside of the smart card's chip, to see if we could find any sensitive areas.

Our initial attack used the following parameters:

- x,y-scan: 5x5 measurements over a 4x4 mm area
- EM-FI Probe power settings: Range, 50% to 85% in steps of 5%.
- VCC Voltage: Fixed, 5V
- Clock Voltage High: Linked to VCC voltage
- Clock Voltage Low: Fixed, 0V
- Wait cycles: Range, 1 to 1,000 in steps of 1
- Glitch cycles: Fixed, 1 pulse
- Glitch offset: Random, 0 to 500 nanoseconds
- Glitch length: Fixed, 20 nanoseconds
- Repetitions: 5

The initial attack resulted in $875,000$ measurements, but it did not lead to any successful of interesting glitches. We did find a boundary area, which contained a scattering of both normal traces and muted traces, but this did not lead to a clear distinction for specific parameter values which might result in a successful glitch.

Focused Attack As our initial scan did not give anything tangible, we chose to do a brute force style search of the boundary area we found. For the brute force search we used the following parameters:

- x,y-scan: 3x3 measurements over a 2x2 mm area
- EM-FI Probe power settings: Range, 75% to 85% in steps of 1%.
- VCC Voltage: Fixed, 5V
- Clock Voltage High: Linked to VCC voltage
- Clock Voltage Low: Fixed, 0V
- Wait cycles: Range, 340 to 350 in steps of 1
- Glitch cycles: Range, 1 to 5 pulses
- Glitch offset: Range, 0 to 1000 nanoseconds in steps of 2 ns
- Glitch length: Fixed, 20 nanoseconds
- Repetitions: 1

Even though we obtained over 6, 249, 474 measurements, and our brute force search covered almost the entire boundary area, we again failed to find any successful glitches. Surprisingly we also failed to find a more defined boundary between the successful and the muted glitches. Apparently the Type I card does not have such a boundary region, or there is some other factor at play which we could not identify.

Conclusions Although we did an extremely exhaustive brute force search over the majority of the card's surface and due to the fact that we did not manage to obtain even one successful glitch, we believe that this card is secure against this type of transient electromagnetic fault injection.

6.3.10 Type J Card

Identification of Card Characteristics The Type J smart card is a card based on JavaCard OS version 2.2.1 with GlobalPlatform 2.1.1. It is FIPS 140-2 Level 3 rated and has a Common Criteria Evaluation Assurance Level of 5+. It is a contact only card, and is loaded with the same Java application as the other JavaCard OS smart cards. It has the following countermeasures:

- High frequency detector
- High voltage detector
- Low frequency detector
- High temperature detector
- Low temperature detector
- The chip is embedded in epoxy, which completely encapsulates the whole Integrated Circuit (IC). Only micro-wires connecting to the faceplate penetrate the epoxy, connecting to the faceplate interface of the module. Attempts to tamper with the module result in damage to the epoxy, the plastic card, or the metal faceplate (scratches, chips, dents, etc.).

We ran a couple of dry-runs of the application, to see how this card dealt with the delays and jitter. As shown in Figure 6.18 there was almost no jitter between sequential traces. This means that if we find a glitch at a certain time, that the glitch should be significantly easier to reproduce than a similar glitch on cards with large delays of excessive clock jitter.

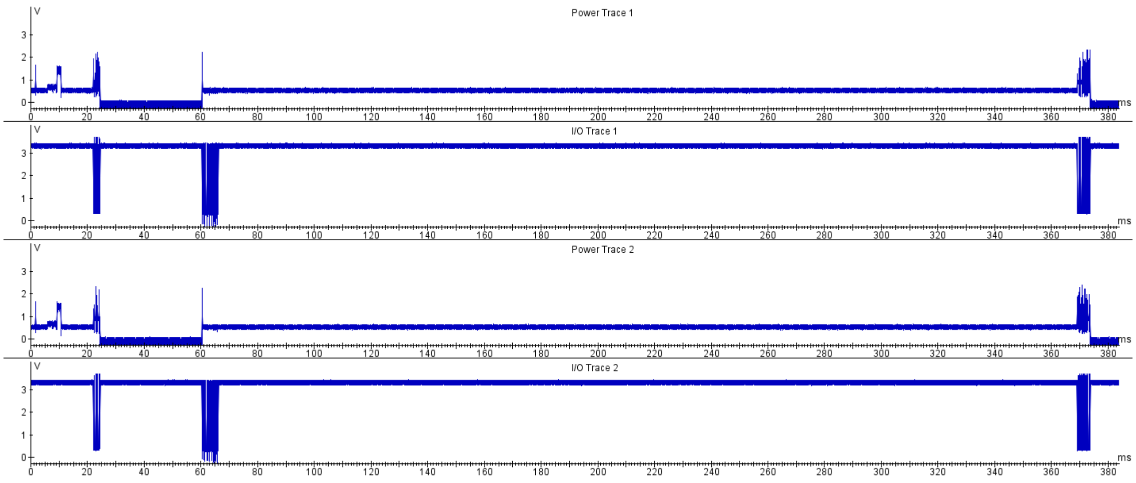


Figure 6.18: A set of 2 power and I/O traces for the Type J smart card, synchronized on the Select ADPU. Note that they are practically identical. This figure is also available full-page in the Appendix as Figure D.11.

Location of Sensitive Areas & Selection of Parameters An x,y-scan was executed on the frontside of the smart card chip using the following parameters:

- x,y-scan: 5x5 measurements over a 4x4 mm area
- EM-FI Probe power settings: Range, 75% to 100% in steps of 5%.
- VCC Voltage: Fixed, 5V
- Clock Voltage High: Linked to VCC voltage
- Clock Voltage Low: Fixed, 0V

- Wait cycles: Random, 1 to 200,000
- Glitch cycles: Range, 1 to 10 pulses
- Glitch offset: Random, 0 to 500 nanoseconds
- Glitch length: Fixed, 20 nanoseconds
- Repetitions: 20

The initial scan led to $30,000$ measurements, including a couple of measurements that were interesting, although no glitches were immediately classified as successful. We decided to focus on one of the boundary areas, where there were a few interesting traces, as well as card mutes and normal results.

Focused Attack The focused attack zoomed in on the following parameters:

- x,y-scan: Fixed, location defined by initial x,y-scan
- EM-FI Probe power settings: Range, 75% to 100% in steps of 5%.
- VCC Voltage: Fixed, 5V
- Clock Voltage High: Linked to VCC voltage
- Clock Voltage Low: Fixed, 0V
- Wait cycles: Range, 27,000 to 28,000
- Glitch cycles: Range, 1 to 10 pulses
- Glitch offset: Random, 0 to 500 nanoseconds
- Glitch length: Fixed, 20 nanoseconds
- Repetitions: 2

The result of the focused attack was $120,120$ traces, of which 2 were considered successful glitches. Beside the two successful glitches, it also became apparent that the Type J card does have a pretty well defined boundary between the normal results and the card mutes. This means that if we want to find more glitch results, that we should zoom in on those areas to find additional parameters that work.

Conclusions Even though the focused attack only gave us two traces, both of which were identical, they do show that the Type J card is sensitive to EM-FI attacks. The glitch result, shown in Listing 6.7 shows that the application can be modified so that it continues to loop more often than necessary.

```
...40 08 00 02 03 E9 FF EC 90 00 23 (Glitched outer loop after 1001 loops)
```

Listing 6.7: Successful glitch result on a Type J smart card

This means that a similar glitch could also be used to break a software based cryptographic algorithm by causing the algorithm to do additional rounds which may render the encryption insecure. Due to the fact that glitching is possible on this card, we must state that we cannot trust the Type J card, as sensitive information may be leaked or security measures circumvented through EM-FI attacks.

6.4 Conclusions

Once we finished all the tests for each of the cards, we made a simple table with an overview of which cards were and which cards were not sensitive to EM-FI. This table is shown in Table 6.3.

Type	Language[a] / OS[b]	Certifications[c d]	Sensitive to EM-FI
Type A	C code	-	Yes
Type B	JC 2.1.1, GP 2.0.1	-	No
Type C	JC 2.1.1, GP 2.0.1	-	Yes
Type D	JC 2.2.1, GP 2.1.1	CC EAL 4+, FIPS 140-2 Level 3	No
Type E	JC 2.2.1, GP 2.1.1	CC EAL 4+	No
Type F	JC 2.2.2, GP 2.1.1	CC EAL 5+	Yes
Type G	JC 2.2.2, GP 2.1.1	CC EAL 5+	Yes
Type H	JC 2.2.1, GP 2.1	CC EAL 5+, FIPS 140-2 Level 3	Yes
Type I	JC 2.2.2, GP 2.1	CC EAL 5+, FIPS 140-2 Level 3	No
Type J	JC 2.2.1, GP 2.1.1	CC EAL 5+, FIPS 140-2 Level 3	Yes

[a]JC: JavaCard version
[b]GP: GlobalPlatform version
[c]CC EAL: Common Criteria Evaluation Assurance Level
[d]FIPS: Federal Information Processing Standard

Table 6.3: The final results for the different types of smart cards tested using EM-FI.

As we can see from the table, cards with extensive countermeasures and cards without countermeasures are both vulnerable to EM-FI. The high Commons Criteria Evaluation Assurance Levels do not mean that a card is safe from EM-FI, nor do the FIPS 140-2 ratings have an effect on whether the card is secure to EM-FI or not. Going purely by the table, one could almost draw the conclusion that CC EAL 4+ cards are safe from EM-FI while CC EAL 5+ cards are not, but this is not the case as CC EAL 5+ envelopes the same criteria as CC EAL 4+ and adds on to that.

It is also remarkable that the Type F and Type G cards are vulnerable to EM-FI while their predecessors the Type D and Type E cards respectively are not vulnerable. In the case of the Type H and Type I cards, the Type I is the successor of the Type H card, and it indeed “fixes” the problem of the older generation card being sensitive to EM-FI. Whether this was by chance or design is not clear. The Type I card does have significantly more countermeasures than the Type H card, but none of those countermeasures are specifically made to counter EM-FI, so we suspect that it is by chance that the card tested invulnerable. Perhaps with additional testing using a smaller probe or other parameters we might still manage to break the insensitive cards.

The biggest conclusions we can draw from the experiments we’ve done in this chapter, are that EM-FI definitely works, and allows for a multitude of different glitches and effects. We also learned that the location of the glitch, as well as the parameters and timing are extremely dependent on the card, it’s architecture, it’s programming and the way the code is put on the card. The JavaCard OS transforms the Java applications to a special type of bytecode, which it then executes on the cards. As we do not know how this code appears it remains difficult to say exactly what part of the software is being glitched, causing some randomness in the successful glitches.

We also noticed that the cards that have both contact and contactless interfaces were slightly better resistant against EM-FI, though this is probably caused mainly by the error correcting and power filtering systems in those cards to prevent the RFID signals from interfering with the cards operation. Last but not least, we noticed that ambient effects, such as the temperature and humidity in the direct vicinity of the experimental setup had an affect on the measurements and results. After a few hundred measurements in a row, the smart cards tended to heat up slightly, often giving a bit more leeway in getting successful glitches.

Chapter 7

Comparison of Fault Injection Techniques

7.1 Fault Injections: A comparison of techniques

Out of all the common fault injection techniques currently available, EM-FI is most closely related to Optical-FI when it comes to the actual methods, the various parameters, and the effect of the attack and the type of glitch that can be produced. This relation is described below.

The experimental setups required for EM-FI and Optical-FI are almost identical. The only real difference between the two is that EM-FI mounts an electromagnetic probe above the smart card chip while Optical-FI typically mounts a laser that focuses its light on the smart card chip through an objective. Generally the EM-FI system is simpler to set up and calibrate due to the fact that it is a relatively coarse attack strategy. The Optical-FI system needs to be calibrated extensively; the objectives need to be focused, the distances need to be fine-tuned and everything needs to be aligned to get the maximum power on the chip instead of heating up the objective or the laser itself.

Additionally the Optical-FI system requires the smart cards to be decapped, so that the light can actually reach the chips' surface. Once the measurements start, the parameters used are almost exactly the same; the biggest difference there is the fact that EM-FI can scan in three dimensions while laser-FI can only scan over the X and Y axes; scanning in the height is pointless when using a focused laser beam. When using Laser-FI it is possible to change the wavelength of the laser, to reach a different depths in the chip, which is unique to laser-FI and conversely EM-FI can change the dimensions of the probe-tip, giving it a different sized area of effect and a change in power.

The effects of the attack for EM-FI and Laser-FI are also very similar. Both techniques target the transistors in a chip, Laser-FI does so by causing a cascade of electrons in the P/N-junction to toggle the transistor while EM-FI modifies the voltage over the transistor's gate which also causes the transistor to toggle. Both Laser and EM appear to be able to make both transient and (semi-)permanent glitches, the laser by increasing the power to cause a transistor to latch-up in a fixed state and EM by giving the transistors an apparent bias[1], which can be reset, as we saw with the Type A card at the end of Section 6.3.1.

The remainder of this chapter focuses on a comparison between EM-FI and the Laser based variant of optical fault injection. In order to give an objective overview of the differences between the two methods, we decided to do an experiment similar to the experiments done for EM-FI

[1]The cause of this phenomenon could be an interesting topic for further research

in Chapter 6. These experiments were done only using the Type A and Type F cards, as it was deemed too expensive to repeat the experiments with all the card types when the two cards should give us a good basis for comparisons.

7.2 Laser-FI Experimentation

Our experiments using the Laser-FI setup were done using a setup practically identical to the EM-FI setup described in Section 6.1, the only major difference was that the EM Probe was replaced by a microscope and 808 nm (red) diode laser, to allow us to focus the laser on the smart cards chip. All other hardware such as the VCGlitcher, PC software and oscilloscopes were identical for both setups. We decided to test the Type A and Type F cards, as they are significantly different cards, giving us the widest sample set without requiring the testing of additional card types. The Type A card does not have any countermeasures and is C-code based, the Type F card does have an extensive list of countermeasures and is JavaCard OS based.

7.2.1 Procedure

The procedure for doing a measurement using Laser-FI is a bit more complicated than when using EM-FI, due to the fact that the card must be decapped first. Decapping can be done in a number of different ways, the goal is always the same, to gain direct line-of-sight access to the front or backside of the chip itself, without destroying the chip in the process.

Once the chip has been decapped the rest of the procedure is extremely similar to EM-FI. First an ordinary execution is observed on the oscilloscope, to gain a power and I/O trace, so we can get an idea of the timing of the attack. We then do an x,y-scan over the surface of the chip, as was the case for EM-FI, using the laser at a low power setting to avoid accidentally triggering the latch-up effect.

One thing to note here, is that in some cases the logic on a chip is visible to the attacker, and certain areas are recognizable as being RAM, ROM, a cache or a processing core. This helps to narrow down the search space significantly, however due to the spot-size Laser-FI still requires a much higher sampling rate than EM-FI to adequately cover the same areas. Once we have found an area that looks promising we can then focus on modifying the parameters so that we obtain glitches that are considered useful or interesting in that area.

7.2.2 Type A card

The first card we attacked using the Laser-FI setup was the Type A card. This was an identical card to the cards used in Section 6.3.1. As the Type A card does not have any hardware countermeasures enabled we decided to decap it mechanically, using a scalpel and a lab technician from Riscure[2] that is skilled at decapping smart cards and embedded chips. The technician stated that the Type A card is best decapped from the frontside of the smart card, through the contact pads, so we asked him to decap one example of a Type A card. Once the card was decapped, it failed to respond to any input whatsoever. Further analysis showed that the chip was sufficiently damaged during the decapping procedure that it ceased functioning. We then asked the technician for a second decapped example, which he then painstakingly decapped and delivered to us. We carefully placed the decapped card in our experimental setup and executed a set of traces to observe the timing and jitter properties.

[2]This technician decaps smart cards almost daily, thus we believe that the damage done during decapping was not caused by inexperience, but rather by the complexity of the task.

We initiated a full x,y-scan over the surface of the chip, with a relatively sparse sample size of 20x20 samples. The other parameters were set similarly to the settings in Section 6.3.1. The laser power was ranged between 0% and 30%. This leads to 300 measurements per grid coordinate for a total of 120, 000. After approximately 66, 000 measurements the card stopped responding to all input. We did find some possibly interesting coordinates in the obtained results, at power levels of 25% and higher, so we decided to do another attempt with a new card, this time restricting the lasers power to a range from 10% to 25%. Our lab technician managed to decap a third Type A card successfully after which we restarted the x,y-scan of the card. After 64, 000 measurements the x,y-scan of the second decapped card was completed. This resulted in 37 successful glitches from which we chose the location with the most successful glitches as the focus for the remainder of the tests. In the focused test, approximately 88 of the 2000 traces were considered successful. This was enough proof that the Type A card is sensitive to Laser-FI.

7.2.3 Type F card

The second card we attacked with the Laser-FI setup was the Type F card. This card, used in Section 6.3.6, has a Common Criteria EAL 5+ rating and has the following countermeasures: low and high clock frequency sensor, low and high temperature sensor, low and high supply voltage sensor, Single Fault Injection (SFI) attack detection, light sensors (included integrated memory light sensor functionality), electronic fuses, active shielding, and a clock input filter for protection against spikes. Of those countermeasures, the SFI, light sensors and active shielding are most efficient against Laser-FI.

For this test we once again needed to decap the smart card. Our lab technician advised us to decap the Type F card from the backside of the card, to expose the frontside of the chip. We managed to destroy 4 cards through both mechanical and chemical decapping methods before we successfully decapped the first card which we could still use for the actual testing procedure. During testing we were planning on running an x,y-scan over the chip's surface to find any sensitive areas, just as we had done for the Type A card, however the decapped card was destroyed when the third light pulse from the laser hit the chip. This laser pulse was only at 5% power and should not have caused any problems, but apparently it triggered a light sensor or somehow rendered the chip incapable of normal operations.

We decided to do a second attempt, and an additional 2 smart cards were destroyed before we managed to successfully decap a second example of the Type F card. This second decapped card survived the initial testing using the laser, but when we started doing a focused attack on a promising set of coordinates the card once again stopped functioning. A total of 8 Type F cards had been destroyed with only an initial scan and no concrete results to show for it. Due to the cards' death-to-results ratio we decided to stop further Laser-FI testing on the Type F card.

7.2.4 Conclusion

In conclusion, from the experiments using both the Type A and Type F cards we learned that Laser-FI has a tendency to destroy smart cards at a pretty high rate, causing it to be a relatively expensive technique to use in practice.

7.3 EM-FI versus Laser

Both EM-FI and Optical-FI have their advantages and disadvantages over each other. The next section contains a list of some of the advantages and disadvantages for both EM-FI and Optical-

FI that were found during the testing in Chapter 6, Section 7.2, and from research done for Sections 4.3 and 4.4. The list is followed by a section of conclusions based on the items in this list.

7.3.1 Advantages EM-FI

- EM-FI does not require for the smart cards to be decapped. The device does not show any physical signs of tampering whatsoever.
- EM-FI also works on decapped targets, with the only significant difference being that less power is required. No additional changes are required.
- On average EM-FI requires fewer cards to do a successful attack as Laser FI tends to destroy cards both during decapping and during glitching, due to the overloading or burning of transistors in the card leading to its destruction.
- EM-FI allows more glitched traces per card before card dies/mutes. Laser tends to burn the surface of the card if too many traces are done in a short time, EM-FI does not suffer from this limitation.
- Capable of glitching multiple (≥ 2) parallel processes simultaneously. Some countermeasures work by doing the same computation on two separate computational cores simultaneously and then comparing the results. EM-FI can cause glitches in both cores simultaneously. Laser-FI would require an additional laser for each individual core that needs to be glitched at the same time.
- EM-FI typically does not trigger certain countermeasures such as light sensors and numerous types of wire-mesh intrusion sensors.
- EM-FI is less sensitive to initial positioning. Finding a location to attack is relatively simple and can usually be done by simply scanning over a 5x5 or 10x10 grid to find a set of coordinates that looks promising. However, relocating the same place (for reproducibility) is difficult due to the surface area of the EM Probe and the inaccuracy caused by the induced EM field.
- EM-FI is typically quicker to give an interesting result than when using Laser, provided the device layout is unknown and both the EM-FI and Laser-FI setup are doing scans of the same resolution etcetera.
- Due to the area-of-effect of the EM probe, in most cases you can simply swap the chip under the probe with another example and reproduce a similar glitch using similar settings, without having to relocate a specific transistor or feature on the chip. If the layouts of the chips are the same, and they are positioned identically, then the chance of hitting the same feature again on the second chip is high.
- In some cases cards go into a permanently muted or lock-down state if it has detected too many incorrect attempts to access the sensitive data. Depending on the card and the manner in which such a state is enforced, certain card states (such as halted or permanently muted) may be reversible by glitching the flags that enforce such a state during the cards power-up phase.[3]
- EM-FI is often successful on chips and devices where Laser fault injection attacks fail. Either by circumventing traditional Optical-FI countermeasures or by glitching multiple features simultaneously so that the device doesn't realize it is being attacked.
- EM-FI can scan over an entire chip in three dimensions (in X, Y and Z) to identify different sensitive areas. Often each of these areas gives unique types of glitches. Laser is limited to

[3]Further research is required. This is theoretically also possible using Laser-FI although the chip might be damaged or destroyed before the correct location and timing are found.

two dimensions (X and Y) and can only reach either the frontside or the backside of the chip, unless the laser is capable of modifying its frequency in such a way that it can shine into the chip.

7.3.2 Advantages Laser

- Laser has a focused target it affects while EM-FI has a relatively large area of effect. The laser's spot size dimensions are usually measured in the range of 500-1000[4] nanometers while the EM probe's spot size is measured in micrometers or even millimeters. A focused spot is preferred if a single specific transistor is to be targeted as the feature sizes on current chips are falling far below the 100 nm range.
- Laser has a significant advantage if the exact layout of the chip is known. This means that there is a smaller target area due to the laser's targeting accuracy it often gives quicker results. EM-FI can utilize the same information about the chip layout, but often knowing the location of a certain component does not really help to find a good attack vector, and you still end up having to do an x,y-scan to find a sensitive location.
- Due to the fact that laser is an optical fault injection technique, you can tell what you are hitting because it is physically visible during frontside attacks. Note that backside laser FI has the same disadvantage as EM-FI; you don't know exactly what you are hitting without further testing.
- Another advantage of Laser-FI is that once the laser has been correctly positioned over a target point on the chip most traces are very reproducible. EM-FI is less accurate, making it more difficult to reproduce an earlier glitch using identical settings. The cost of this feature is that initial placement of the laser is a lot more time consuming than it is for EM-FI.
- Laser is accurate enough to hit targets in areas that are immediately surrounded by sensitive areas (such as ROM or EEPROM) which might get cleared/corrupted by EM fields.
- Laser is quicker and more accurate then EM-FI when the target is already decapped and does not have laser-specific countermeasures such as light sensors. EM-FI is less accurate in small areas and can accidentally trigger the countermeasures or hit components that should not be hit, such as memory cells and caches.

7.4 Conclusions

During the experiments for Chapter 6 and Section 7.2 we noticed a large number similarities between the Laser-FI techniques and EM-FI techniques, but we also noticed a couple of significant differences.

The basic technique for an attack using EM-FI is very similar to an attack using Laser-FI. In both cases the chip is scanned in a simple x,y-scan in order to locate any areas that are sensitive to the method of attack. When using Laser-FI it is possible, assuming the layout of the chip is known, to focus the attack on a certain area to avoid hitting certain parts of the chip and potentially triggering countermeasures or modifying critical memory regions. EM-FI does not allow such fine-tuning and thus, in most cases, it is easier to simply scan the entire chip surface to find a location that is vulnerable to the electromagnetic fields.

Once a location has been found, the parameters that must be selected are practically identical for both the Laser-FI and EM-FI setups, with the parameters such as the timings and power

[4]This is limited by the physics of light. The spot can not have smaller dimensions than the wavelength of its photons [55]

usage for the laser/EM coil being the most important. Both setups start sending pulses at a low power level, to avoid burning or short-circuiting the chip, and slowly build up to a level where it causes glitches without destroying the features. Laser does have a tendency to burn-in and cause latch-ups, meaning that if the same location is glitched repeatedly, it will at one point cause a burn in the silicon or cause a latch-up effect and may not continue to function as well afterwards.

The biggest difference between the two is the accuracy of the actual attack. EM-FI can be compared to a shot-gun approach while laser-FI is comparable to a surgical scalpel; both techniques work to attack the target, but the basic philosophy is entirely different. EM-FI tends to hit a large amount of features on the chip at the same time, often causing voltage spikes in multiple lines and causing multiple transistors to toggle or multiple bits to flip in memory. Laser-FI has very fine accuracy, so it can cause only a single transistor to toggle, if the feature size of the chip allows.

We also noticed that Laser-FI testing tends to destroy a lot more smart cards than testing using EM-FI. Where EM-FI usually only requires one or two cards for testing, Laser-FI often needs 10 or more to do a complete test of a type of card. EM-FI produces transient faults, so in most cases after a hard reset any changes the glitching has produced are simply cleared and the chip will resume normal operations. Only in cases where the chip detects the attack and triggers a destructive protection mechanism, or where a glitch modifies the smart card's programming in such a way that it can no longer boot, does EM-FI destroy the smart card. There have been cases where a glitch caused a smart card chip to start using a different ATR, meaning that the cards' ROM was modified, but even in those cases the card continued to function. Laser-FI on the other hand requires decapped chips to do an attack. However as most smart cards are protected against tampering, it is difficult to successfully decap a smart card chip without destroying the smart card in the process.

As an example: During our testing in Section 7.2 we managed to destroy a total of 6 Type F cards during the decapping phase and an additional 2 cards during the laser testing. Using EM-FI however, we managed to do 2 successful fault attacks (approximately 1.5 million glitched measurements), including initial scans and focused attacks, before the first Type F card stopped responding. The second Type F card we attempted to glitch showed no signs of wear even after 4.5 million glitched executions. In comparison, a colleague also attacked a Type F card using VCC glitching; VCC glitching was less destructive than the Laser-FI attacks, but also managed to destroy 3 samples of the card during his testing. Similarly in the case of the Type A smart card, Laser-FI and the decapping process were responsible for 3 cards being destroyed while EM-FI was only responsible for the death of a single Type A card. The card killed by EM-FI had a modified boot ROM so it might even be possible to recover the card using a boot flashing utility.

Due to the big difference in the number of cards required, as well as the fact that EM-FI triggered far less countermeasures than the Laser-FI setup did, we believe that EM-FI is preferable over Laser-FI. This is especially the case for chips with unknown countermeasures, chips with a limited number of samples, and chips for which the layout is unknown. Likewise if the smart cards must remain unmodified and they are protected against VCC glitching, they can not be decapped and VCC glitching will fail, in which case EM-FI becomes the only viable option. Similarly if the chip has countermeasures against Optical-FI, such as light sensors, it becomes incredibly difficult to successfully decap and glitch a chip. Most countermeasures against Optical-FI will not work against EM-FI, meaning that in almost every case, EM-FI has the advantage.

Chapter 8

Final Conclusions

The goal of this thesis was to find an answer to the question:

> How does EM-FI compare to other fault injection methods, such as Optical-FI, with respect to testing techniques and sensitivity to countermeasures?

We wanted to know what the advantages and disadvantages of EM-FI are with respect to existing FI methods, to help establish the impact EM-FI can have on the state of current fault injection techniques and the ongoing protection against those same fault injection techniques. Smart cards and embedded devices feature a broad range of countermeasures and are often certified with different classifications that show that the chips are protected against certain types of attacks, but little was known about how those countermeasures effect the chips' sensitivity to EM-FI. We asked ourselves: "Is EM-FI the next best thing for attackers and the nightmare of chip producers, or is EM-FI too small in the scope of things to really matter?"

As it turns out, EM-FI is a major player in the field of fault injection techniques, and it is definitely here to stay.

The idea behind the research question could be solved by dividing it into three separate points of interest, leading to the sub questions:

1. What testing approach is needed when using EM-FI, and what kind of effects are possible?
2. Are common fault injection protection mechanisms and countermeasures, such as countermeasures against Optical-FI, effective against EM-FI or does the emergence of EM-FI require additional security measures, testing and certifications for smart cards?
3. What are the advantages and disadvantages of EM-FI versus Optical-FI?

In Chapter 6 we described the approach needed when doing EM-FI attacks and also gave various examples of the types of glitches that were possible, practically answering Question 1. We simultaneously answered Question 2 about whether common fault injection protection mechanisms and countermeasures are sufficient to protect chips against EM-FI. As we can see from Table 8.1, taken from the conclusion of Chapter 6 (Section 6.4), even the cards that were CC EAL 5+ rated and had long lists of countermeasures were still found to be vulnerable to EM-FI attacks.

Judging by the results in Table 8.1, the conclusions in Section 6.4, as well as the knowledge that the current techniques are still being improved, we strongly recommend chip manufacturers to start designing and implementing protection mechanisms and countermeasures against EM-FI. Countermeasures such as voltage sensors, temperature sensors and light sensors can and will help against alternative fault injection techniques, but EM-FI is capable of avoiding the current countermeasures and thus it is a wide open door for attackers. Additionally it would probably be a good idea to take a good look at the certifications, such as the FIPS 140-2 and CC EAL ratings, to perhaps expand those to cover EM-FI related attacks as well as the existing fault injection techniques already covered.

Type	**Language**[a] **/ OS**[b]	**Certifications**[c d]	**Sensitive to EM-FI**
Type A	C code	-	Yes
Type B	JC 2.1.1, GP 2.0.1	-	No
Type C	JC 2.1.1, GP 2.0.1	-	Yes
Type D	JC 2.2.1, GP 2.1.1	CC EAL 4+, FIPS 140-2 Level 3	No
Type E	JC 2.2.1, GP 2.1.1	CC EAL 4+	No
Type F	JC 2.2.2, GP 2.1.1	CC EAL 5+	Yes
Type G	JC 2.2.2, GP 2.1.1	CC EAL 5+	Yes
Type H	JC 2.2.1, GP 2.1	CC EAL 5+, FIPS 140-2 Level 3	Yes
Type I	JC 2.2.2, GP 2.1	CC EAL 5+, FIPS 140-2 Level 3	No
Type J	JC 2.2.1, GP 2.1.1	CC EAL 5+, FIPS 140-2 Level 3	Yes

[a]JC: JavaCard version
[b]GP: GlobalPlatform version
[c]CC EAL: Common Criteria Evaluation Assurance Level
[d]FIPS: Federal Information Processing Standard

Table 8.1: The final results for the different types of smart cards tested using EM-FI, copied from Table 6.3 in Section 6.4.

This also ties in to the last question we answered in this thesis, namely Question 3, the advantages and disadvantages of EM-FI versus Optical-FI. The extensive list of pros and cons, as listed in Chapter 7 gives a good overview of both techniques. When we couple that with what we learned from the rest of this thesis, we can only come to one conclusion: Until chip manufacturers start implementing countermeasures against EM-FI, making it significantly more expensive and difficult to perform glitches using that technique, EM-FI is superior to all other fault injection techniques currently available. Especially so for cards that already have existing countermeasures against other forms of SCA and FI.

So putting it all together, we can finally answer our original research question in a simple to the point manner:

> How does EM-FI compare to other fault injection methods, such as Optical-FI, with respect to testing techniques and sensitivity to countermeasures?

To put it bluntly, EM-FI is better than the alternatives such as Laser-FI. It is simpler, significantly less expensive, and a lot less sensitive to the existing countermeasures on most of today's smart cards and embedded devices.

Chapter 9

Future Work

Smart card technology is ever changing, and with the emergence of successful EM-FI techniques, countermeasures against EM-FI will be appearing in chips soon. In order to keep up with the advances in technology, there are a number of additional projects that can be looked in to to improve both the smart card chip technology as well as to improve the ease-of-use and reliability of the EM-FI setup that we used.

In order to improve the use of the EM-FI setup, it might be a good idea to use Riscure's ICWaves device to optimize timing accuracy. The ICWaves is an advanced pattern-based triggering device for generating time independent pulses to avoid jitter and time-related countermeasures in SCA or FI testing [46]. This should help to practically remove all timing and jitter issues, making the results significantly more reliable and easier to reproduce. Similarly, the same setup and techniques should carry over into the world of embedded circuits. By using Riscure's Glitch Amplifier [45] it is possible to connect the VC Glitcher [49] to an embedded circuit and use it to test those devices for faults.

One of the biggest disadvantages of using the EM-FI probe is the size of the tips used. We used a prototype, meaning that we had probe tips available ranging from 4 mm diameter down to 1.5 mm diameter, but even the smallest tips cover a large area when positioned over a smart card chip. Future research into decreasing the size of such tips without decreasing the range or power significantly would help to improve the overall experience when using EM-FI. One possible method of doing this would be to place a ferromagnetic cone shaped attachment at the bottom of a probe, causing the electromagnetic flux to be funneled from the 1.5 mm diameter coil into a much smaller tip, at the cost of a bit of power due to self-induction. A smaller tip would help focus the EM field on a smaller area, allowing much more accurate (re-)location of sensitive areas, as well as to avoid ROM or EEPROM areas when glitching chip features in the close vicinity to those areas.

Another topic for potential further research is the effects of the Positive and Negative probe tips. It may be interesting to see why some cards are extremely sensitive to attacks using one probe tip, while they are practically immune to attacks using the other probe tip. Is the effect caused solely by the layout of the wires and features in the chip or are there additional factors that play a role?

Harmonic EM-FI is another area that begs for more research, including a comparison of the effects of transient EM-FI to Harmonic EM-FI, and an overview of which technique is best suited in which type of scenario.

A final interesting bit of research might be to try to recover "dead" or system-halted smart cards by injecting faults during the start-up and ATR phases. Many cards don't do a complete wipe of their memory when they enter a system-halted state and stop responding to input. Often such cards only set a flag that is checked during start-up which tells the card to stop responding to input. If it is possible to glitch the card during such a check it might be possible to make such a glitch permanent and thus resurrect a dead card, allowing for further testing or the retrieval of sensitive information. Similarly often banking cards are blocked after 3 or 5 incorrect PIN number attempts. Perhaps these blocks can also be ignored or reset using fault injection techniques.

Bibliography

[1] Radiation effects in integrated circuits. In *Fault-Tolerance Techniques for SRAM-based FPGAs*, volume 32, pages 9–27. Springer US, 2006. 20

[2] Maurice Aarts. Hardware Attacks: Tamper Resistance, Tamper Response and Tamper Evidence. 2012. [UNPUBLISHED], `http://maurice.aarts.info/papers/tamper_resistance_evidence.pdf`. 23

[3] Ali Alaeldine, Thomas Ordas, Richard Perdriau, Philippe Maurine, Mohamed Ramdani, Lionel Torres, and M'hamed Drissi. Assessment of the Immunity of Unshielded Multi-Core Integrated Circuits to Near-Field Injection. In *Electromagnetic Compatibility, 2009 20th International Zurich Symposium on*, pages 361–364, 2009. 6

[4] Ross Anderson and Markus Kuhn. Tamper Resistance: a Cautionary Note. In *Proceedings of the 2nd conference on Proceedings of the Second USENIX Workshop on Electronic Commerce-Volume 2*, pages 1–11. USENIX Association, 1996. 5

[5] Ross Anderson and Markus Kuhn. Low cost attacks on tamper resistant devices. *Security Protocols*, pages 125–136, 1998. 5

[6] Anonymized. Smart Card Specifications Document. Anonymized to prevent card alias from becoming common knowledge. Contact Riscure B.V. for more details. 28, 29, 31, 32, 33

[7] Pierre Bayon, Lilian Bossuet, Alain Aubert, Viktor Fischer, François Poucheret, Bruno Robisson, and Philippe Maurine. Contactless electromagnetic active attack on ring oscillator based true random number generator. In *Constructive Side-Channel Analysis and Secure Design*, pages 151–166. Springer, 2012. 6, 21

[8] IBM BlueZ. JCOP Tools 3.0 (Eclipse Plugin). Technical report, IBM, 2012. Rev. 1.0. 37

[9] Rafael José Boix Carpi. Optimization of parameter settings search for a successful Fault Injection. Master's thesis, Universitat Politècnica de València. Escuela Tècnica Sup. de Ingeniería Informática-Escola Tècnica Sup. d'Enginyeria Informàtica, 2013. 40, 42, 101

[10] D. Boneh, R. DeMillo, and R. Lipton. New Threat Model Breaks Crypto Codes. Bellcore Press Release, September 25th 1996. 5

[11] Dan Boneh, Richard A. Demillo, and Richard J. Lipton. On the importance of checking cryptographic protocols for faults. pages 37–51. Springer-Verlag, 1997. 9

[12] G Canivet, P Maistri, R Leveugle, J Clédière, F Valette, and M Renaudin. Glitch and laser fault attacks onto a secure AES implementation on a SRAM-based FPGA. *Journal of cryptology*, 24(2):247–268, 2011. 6

[13] Sebastian Carlier. Electro Magnetic Fault Injection. Master's thesis, UvA - Universiteit van Amsterdam, 2012. 6, 40

[14] Vincent Carlier, Hervé Chabanne, Emmanuelle Dottax, and Hervé Pelletier. Electromagnetic Side Channels of an FPGA Implementation of AES. *IACR eprint archive*, 2004. 5

[15] Amine Dehbaoui, Jean-Max Dutertre, Bruno Robisson, P. Orsatelli, Philippe Maurine, and Assia Tria. Injection of transient faults using electromagnetic pulses -Practical results on a cryptographic system-. *IACR Cryptology ePrint Archive*, 2012:123, 2012. informal publication. 6

[16] Amine Dehbaoui, Jean-Max Dutertre, Bruno Robisson, and Assia Tria. Electromagnetic Transient Faults Injection on a Hardware and a Software Implementations of AES. In Guido Bertoni and Benedikt Gierlichs, editors, *FDTC*, pages 7–15. IEEE, 2012. 6

[17] Paul N. Fahn and Peter K. Pearson. IPA: A New Class of Power Attacks. In *Cryptographic Hardware and Embedded Systems, First International Workshop, CHES'99, Worcester, MA, USA, August 12-13, 1999, Proceedings*, volume 1717 of *Lecture Notes in Computer Science*, pages 173–186. Springer, 1999. 5

[18] Joe Grand. Protecting your crown jewels: an introduction to embedded security for hardware-based products. *Computer Fraud & Security*, 2005(10):13 – 20, 2005. 26

[19] Y. Hayashi, N. Homma, T. Sugawara, T. Mizuki, T. Aoki, and H. Sone. Non-invasive EMI-based fault injection attack against cryptographic modules. In *Electromagnetic Compatibility (EMC), 2011 IEEE International Symposium on*, pages 763–767, 2011. 6

[20] Yu-ichi Hayashi, Naofumi Homma, Takeshi Sugawara, Takaaki Mizuki, Takafumi Aoki, and Hideaki Sone. Non-invasive Trigger-free Fault Injection Method Based on Intentional Electromagnetic Interference. *Proc. NIAT 2011*, 2011. 6

[21] J. D. Hunter. Matplotlib: A 2D graphics environment. *Computing In Science & Engineering*, 9(3):90–95, 2007. 40

[22] International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC), Geneva, Switzerland. *Identification cards – Contactless integrated circuit(s) cards Proximity cards*. ISO/IEC: 14443-1 (2008), 14443-2 (2010), 14443-3 (2011), 14443-4 (2008). 7

[23] International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC), Geneva, Switzerland. *Identification cards – Integrated circuit cards*. ISO/IEC: 7816-1 (2011), 7816-2 (2007), 7816-3 (2006), 7816-4 (2013), 7816-5 (2004), 7816-6 (2004), 7816-7 (1999), 7816-8 (2004), 7816-9 (2004), 7816-10 (1999), 7816-11 (2004), 7816-12 (2005), 7816-13 (2007), 7816-15 (2004). 7, 8, 9, 13

[24] International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC), Geneva, Switzerland. *Identification cards – Physical characteristics*. ISO/IEC: 7810 (2003). 7

[25] Ognjen Jović. Susceptibility of ICs to Conducted Electromagnetic Interference. 6

[26] Chong Hee Kim and J. J Quisquater. Faults, Injection Methods, and Fault Attacks. *Design Test of Computers, IEEE*, 24(6):544–545, 2007. 5

[27] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential Power Analysis. In *Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '99, pages 388–397, London, UK, UK, 1999. Springer-Verlag. 5

[28] François Koeune and François-Xavier Standaert. A Tutorial on Physical Security and Side-Channel Attacks. In Alessandro Aldini, Roberto Gorrieri, and Fabio Martinelli, editors, *Foundations of Security Analysis and Design III*, volume 3655 of *Lecture Notes in Computer Science*, pages 78–108. Springer Berlin Heidelberg, 2005. 5

[29] Mehdi laurent Akkar, Louis Goubin, Olivier Ly, and Universit Bordeaux I. Automatic Integration of Counter-Measures Against Fault Injection Attacks, 2009. 5

[30] Xuefei Leng. Smart card applications and security. *Information Security Technical Report*, 14(2):36 – 45, 2009. Smart Card Applications and Security. 24

[31] Philippe Loubet-Moundi, David Vigilant, and Francis Olivier. Static fault attacks on hardware des registers. *IACR Cryptology ePrint Archive*, 2011:531, 2011. 9

[32] Ken Mai. Side channel attacks and countermeasures. In Mohammad Tehranipoor and Cliff Wang, editors, *Introduction to Hardware Security and Trust*, pages 175–194. Springer New York, 2012. 25

[33] Martin Otto. *Fault Attacks and Countermeasures*. PhD thesis, University of Paderborn, 2005. 5, 14

[34] Philippe Maurine. Techniques for EM Fault Injection: Equipments and Experimental Results. In *FDTC'2012: Fault Diagnosis and Tolerance in Cryptography*, pages 003–004, 2012. 6

[35] David Oswald. *Development of an Integrated Environment for Side Channel Analysis and Fault Injection*. PhD thesis, Ruhr-Universität Bochum, 2009. 6

[36] Eric Peeters, François-Xavier Standaert, and Jean-Jacques Quisquater. Power and electromagnetic analysis: Improved model, consequences and comparisons. *Integration, the {VLSI} Journal*, 40(1):52 – 60, 2007. Embedded Cryptographic Hardware. 5

[37] Pico Technology. *PicoScope 5000 Series*, 2012. 36

[38] F. Poucheret, K. Tobich, M. Lisarty, L. Chusseau, B. Robisson, and P. Maurine. Local and direct em injection of power into cmos integrated circuits. In *Fault Diagnosis and Tolerance in Cryptography (FDTC), 2011 Workshop on*, pages 100–104, 2011. 21

[39] Jean-Jacques Quisquater and David Samyde. Electromagnetic analysis (EMA): Measures and counter-measures for smart cards. In *Smart Card Programming and Security*, pages 200–210. Springer Berlin Heidelberg, 2001. 5, 29, 30

[40] W. Rankl and W. Effing. *Smart Card Handbook*. John Wiley & Sons, Inc., New York, NY, USA, 3 edition, 2003. 7

[41] Wolfgang Rankl. Overview about attacks on smart cards. *Information Security Technical Report*, 8(1):67 – 84, 2003. 26

[42] Riscure B.V. *Diode Laser Station Datasheet*, 2012. 15, 16

[43] Riscure B.V. EM-FI R&D Report v0.5. Technical report, Riscure B.V., 2012. [Confidential/Unpublished]. 6, 17

[44] Riscure B.V. *EM Probe Station Datasheet*, 2012. 36

[45] Riscure B.V. *Glitch Amplifier Datasheet*, 2012. 81

[46] Riscure B.V. *IC Waves Datasheet*, 2012. 81

[47] Riscure B.V. *Inspector v4.6 Datasheet*, 2012. v, 37

[48] Riscure B.V. *Inspector v4.6 SCA and FI Tool - Software & User Manual*, 2012. xv, 11, 13

[49] Riscure B.V. *VC Glitcher Datasheet*, 2012. 36, 81

[50] Riscure B.V. *EM-FI Transient Probe Datasheet*, 2013. 16, 19, 20, 36

[51] Jörn-Marc Schmidt. Differential Fault Analysis. Technical report, A report from IAIK Lab in Austria, 2008. 5, 12, 88, 89

[52] Jörn-Marc Schmidt and Michael Hutter. Optical and EM Fault-attacks on CRT-based RSA: Concrete results. In *Proceedings of the Austrochip*, pages 61–67. Citeseer, 2007. 5

[53] Ahmadou A. Sere, Julien Iguchi-Cartigny, and Jean-Louis Lanet. Automatic detection of fault attack and countermeasures. In *Proceedings of the 4th Workshop on Embedded Systems Security*, WESS '09, pages 7:1–7:7, New York, NY, USA, 2009. ACM. 6

[54] Sergei Skorobogatov. Physical attacks and tamper resistance. In Mohammad Tehranipoor and Cliff Wang, editors, *Introduction to Hardware Security and Trust*, pages 143–173. Springer New York, 2012. 24, 25, 28

[55] J.G.J. van Woudenberg, M.F. Witteman, and F. Menarini. Practical optical fault injection on secure microcontrollers. In *Fault Diagnosis and Tolerance in Cryptography (FDTC), 2011 Workshop on*, pages 91–99, 2011. 6, 77

[56] Mathias Wagner. 700+ attacks published on smart cards: The need for a systematic counter strategy. In *COSADE 2012, LNCS 7275*, pages 33–38, 2012. 9

[57] Steve Weingart. Physical security devices for computer subsystems: A survey of attacks and defenses. In Çetin Koç and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems CHES 2000*, volume 1965 of *Lecture Notes in Computer Science*, pages 45–68. Springer Berlin / Heidelberg, 2000. 10.1007/3-540-44499-8_24. 23, 25, 26, 27

Appendix A

Notation

- In accordance with common usage, the term ‘byte’ refers to a sequence of eight bits and is equivalent to the term ‘octet’, which is often used in international standards. Furthermore, in this document the abbreviation ‘kb’ stands for kilobit, and ‘kB’ represents kilobyte. Thus ‘1kB’ = ‘8kb’.
- When referring to bits and bytes, the prefixes ‘kilo’ and ‘mega’ have the values of 1024 (2^{10}) and 1048576 (2^{20}), respectively, as is customary in the field of information technology. When referring to other units of measures, such as but not limited to Hertz, cycles, samples, seconds, etc... the standard metric-system prefixes are used, as defined by the International System of Units (SI)[1].
- Length specifications for data, objects and all countable quantities are shown in decimal form, in agreement with the usual practice in smart card standards. Unless otherwise noted, all other values are shown as hexadecimal numbers and identified as such by denoting them with a ‘0x’-prefix (ie. hexadecimal `0xCAFE` = `0xcafe` = 51966 in decimal notation).
- Mathematical:

  $||$ - Concatenation of two objects or data elements.

  $\oplus$ - Logical Exclusive OR (XOR) operation.

  $\vee$ - Logical OR operation.

  $\wedge$ - Logical AND operation.

  $a \in A$ - a is an element of the set A.

  $a \notin B$ - a is not an element of the set B.

  $\{a, b, c\}$ - The set of elements containing a, b, and c.

  $[a, b, c, c]$ - The bag of elements containing a, b, c, and c.

[1] The list of prefixes: `http://en.wikipedia.org/wiki/Metric_prefix`

Appendix B

Definitions

Passive attack - A passive attack on a cryptosystem is one in which the cryptanalyst cannot interact with any of the parties involved, attempting to break the system solely based upon observed data. This is usually done through monitoring or traffic analysis. Due to their passive nature, passive attacks are very difficult to detect because they do not involve any alteration of the data, and neither the sender nor the receiver are aware that a third party has obtained a copy of the messages.

Active attack - An active attack on a cryptosystem is one in which the cryptanalyst actively injects or modifies traffic between the parties involved. The attacker can also resort to doing a Man-in-the-middle attack or by simply acting like the second party. Active attacks are slightly easier to detect because the attacker has to send data to the other parties, however depending on the protocol used the attacker may be able to make himself indistinguishable from the legitimate parties. Active attacks are usually based on passive attacks, but allow the attacker to interact with the system, to allow more specific data analysis.

Traffic analysis - The eavesdropper analyzes the traffic going to and/or from a device or location. Often all incoming and out going traffic on the network is analyzed for interesting data. The eavesdropper can use this to determine the location of one or more parties, or to identify the communicating hosts. The eavesdropper can also observe the frequency and length of messages being exchanged. By using all these pieces of information they can then predict the nature of the communication without having to alter a single message on the network.

Invasive attack - A physical attack in which the attacker brings irreparable harm or physical damage to the device under test, such that the device shows it has been tampered with, or displays modified functionality. This includes depackaging the embedded chip, removal of passivation layers, drilling, etching, and in some cases the device might be completely disassembled or destroyed. This technique is often used when microprobing or reverse-engineering the device. Such invasive methods establish direct electrical contact to the surface of the chip. These attacks usually need very expensive equipment such as a probe station, a laser cutter, or a focused ion beam (FIB) [51].

Semi-invasive attack - A physical attack in which the attacker brings irreparable harm or physical damage to the packaging of the device under test. The chip's electronic circuit and passivation layers usually remain intact and unmolested, but the device itself shows obvious signs of tampering. This technique is usually used when using techniques that require to be in very close proximity to the chip, but do not require direct contact to the chip surface, such as UV-light, photon-emission, radiation, laser and EM based SCA and FI attacks. The behavior of the device can be influenced without direct electrical contact to the chip. Such attacks usually require sophisticated equipment or materials such as chemicals for the decapsulation procedure [51].

Non-invasive attack - A physical attack in which the attacker does not damage or change the device under test in such a way that the tampering becomes evident. After such an attack, the devices shows no signs that an attack has taken place. This technique only allows SCA and FI methods that work from outside the device, such as power and electromagnetic emission analysis. Faults are provoked by manipulating the operating conditions of the device. This can be done by injecting peaks in to the clock or the power supply, which is called glitch or spike attack, respectively. Another possibility is to increase or decrease the temperature outside of the normal operating conditions. These methods are relatively inexpensive and easy to perform, but offer only a limited precision as these attacks are slow and impact large areas or even the entire chip at once [51].

Direct convention - Logical convention which encodes binary data with high voltage (H) and low voltage (L) to represent the bits. Bits with logic value "1" are transfered using high voltage and bits with logic value "0" are transfered using low voltage, with the least-significant bit of each data byte being transmitted first and the most-significant bit is transmitted last. For example, the byte `0x3B` is transmitted as (H)L-*HHLHHHLL*-H(H), with the (H) indicating the idle (high) state of the I/O line.

Inverse convention - Logical convention which encodes binary data with low voltage (L) and high voltage (H) to represent the bits. Bits with logic value "1" are transfered using low voltage and bits with logic value "0" are transfered using high voltage, with the most-significant bit of each data byte being transmitted first and the least-significant bit is transmitted last. For example, the byte `0x3F` is transmitted as (H)L-*HHLLLLLL*-H(H), with the (H) indicating the idle (high) state of the I/O line.

Trace - An acquisition of the signals or bytes sent to and received from the device under test.

Signal Trace - A data acquisition using an oscilloscope to record the voltages on a certain signal line at a specific frequency. Usually a signal trace refers to a power trace, which is normally displayed as a plot of the power consumed by a device with respect to time. Other trace possibilities include Input/Output (IO) traces, Clock signal traces and EM probe coil traces.

Normal Trace - A trace in which the device under test displays normal operations and returns the expected output.

Interesting Trace - A trace in which the device under test displays modified behavior. This can be either a mute trace, an error trace, a successful glitch trace, or some other result that is unexpected and might be worth looking into.

Mute Trace - A trace in which the device under test stops responding to any in until the next reset. This is a self protection mechanism to complicate fault injection attacks by not returning any more information whatsoever once an attack has been detected.

Error Trace - A trace in which the device under test stops responding to any in until the next reset. This is a self protection mechanism to complicate fault injection attacks by not returning any more information whatsoever once an attack has been detected.

Successful Glitch (Trace) - A trace in which the device under test displays unexpected behavior and returns alternative output that can be utilized in an attack.

Appendix C

Acronyms, Abbreviations & Symbols

ADPU	Application Protocol Data Unit Command
AES	Advanced Encryption Standard
AID	Application Identifier
API	Application Programming Interface
ATM	Automated Teller Machine
ATR	Answer to Reset
CC	Common Criteria
CLA	Class Byte
CPA	Correlation Power Analysis
CRT	Chinese Remainder Theorem
DEMA	Differential Electromagnetic Analysis
DES	Data Encryption Standard
DFA	Differential Fault Analysis
DPA	Differential Power Analysis
DUT	Device Under Test
EAL	Evaluation Assurance Level
EC	Elliptic Curve
EC DSA	Elliptic Curve Digital Signature Algorithm
ECC	Elliptic Curve Cryptography
EEPROM	Electrically Erasable Programmable Read-Only Memory
EM	Electromagnetic
EMA	Electromagnetic Analysis
EM-FI	Electromagnetic Fault Injection
ETU	Elementary Time Unit
FI	Fault Injection
FIPS	Federal Information Processing Standard
FPGA	Field Programmable Gate Array
GSM	Global System for Mobile Communications
GS/s	GigaSamples per second
ICC	Integrated Circuit Card
INS	Instruction Byte
I/O	Input/Output
IEMI	Intentional Electromagnetic Interference
IPA	Inferential Power Analysis

JTAG	Joint Test Action Group
JCOP	Java Card Open Platform
L_c field	Length field encoding the number N_c
L_e field	Length field encoding the number N_e
NC	Not Connected
N_c	Number of bytes in the command data field
N_e	Maximum number of bytes expected in the response data field
N_r	Number of bytes in the response data field
P1-P1	Parameter Bytes
PCB	Printed Circuit Board
Op-FI	Optical Fault Injection
RF	Radio Frequency
RFID	Radio Frequency Identification
RFU	Reserved for Future Use
RID	Registered Application Provider Identifier
RNG	Random Number Generator
SCA	Side-Channel Analysis
SEMA	Simple Electromagnetic Analysis
SoC	System on Chip
SPA	Simple Power Analysis
SRAM	Static Random Access Memory
SW1-SW2	Status Bytes
TLV	Tag, Length, Value
TRNG	True Random Number Generator

Appendix D

Full-page Figures

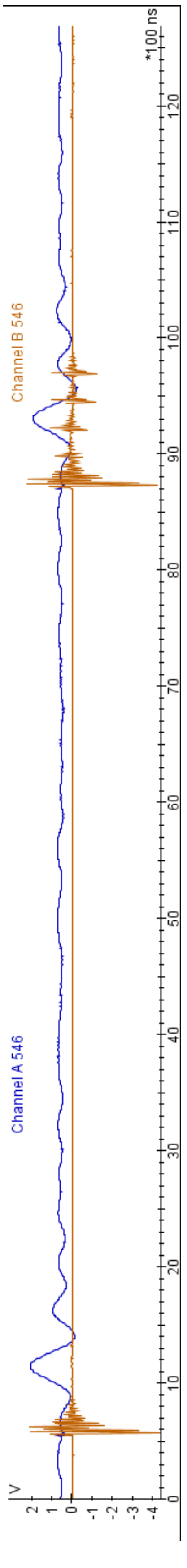


Figure D.1: A Power and EM Coil Trace on a successful glitch of a Type A card. Channel A denotes the Power trace, Channel B is the voltage over the EM Coil.

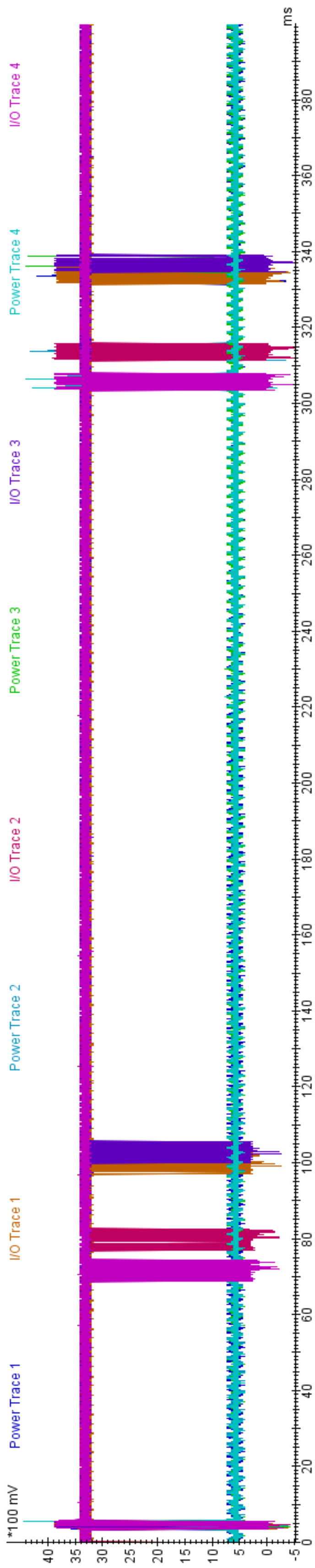


Figure D.2: A set of 4 power and I/O traces for the Type B smart card, synchronized on the Select ADPU.

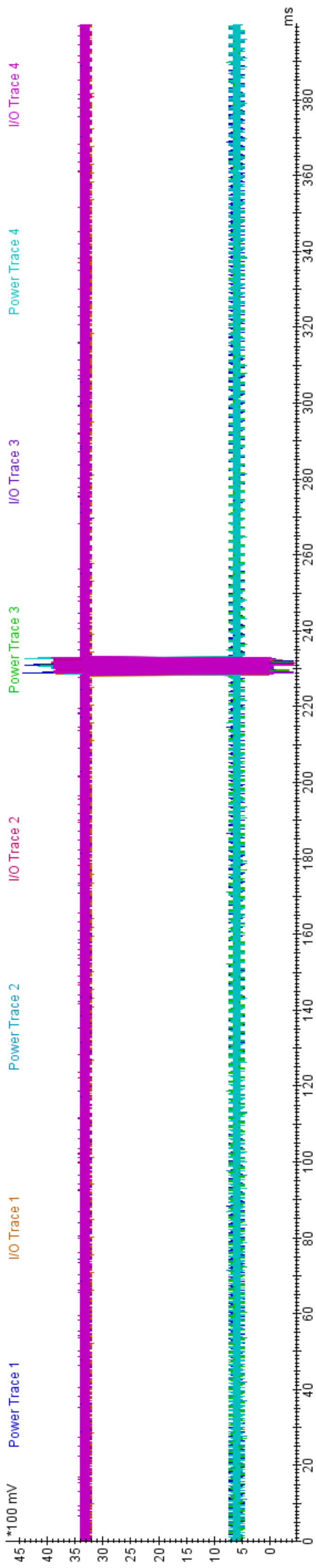


Figure D.3: A set of 4 power and I/O traces for the Type B smart card, synchronized on the Command ADPU.

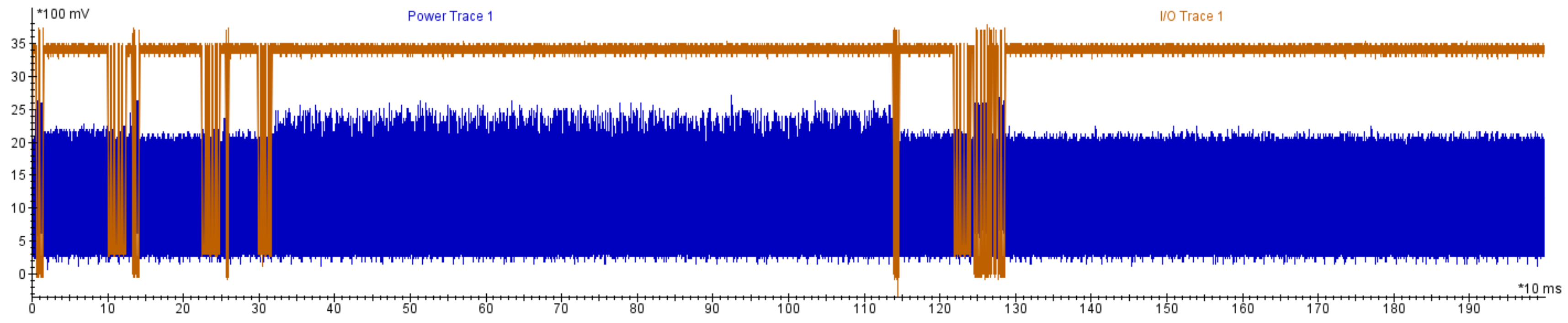


Figure D.4: Power and I/O trace 1 for the Type C smart card, synchronized on the Select ADPU.

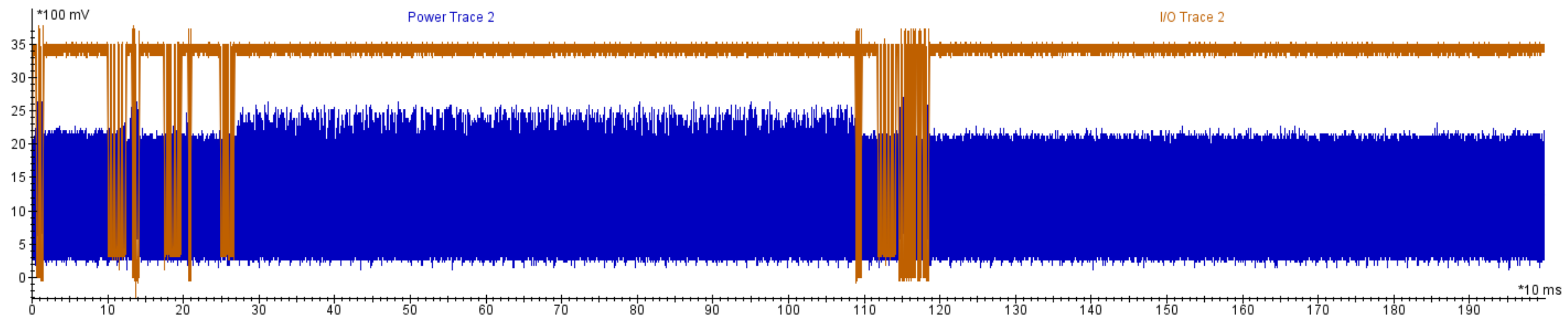


Figure D.5: Power and I/O trace 2 for the Type C smart card, synchronized on the Select ADPU.

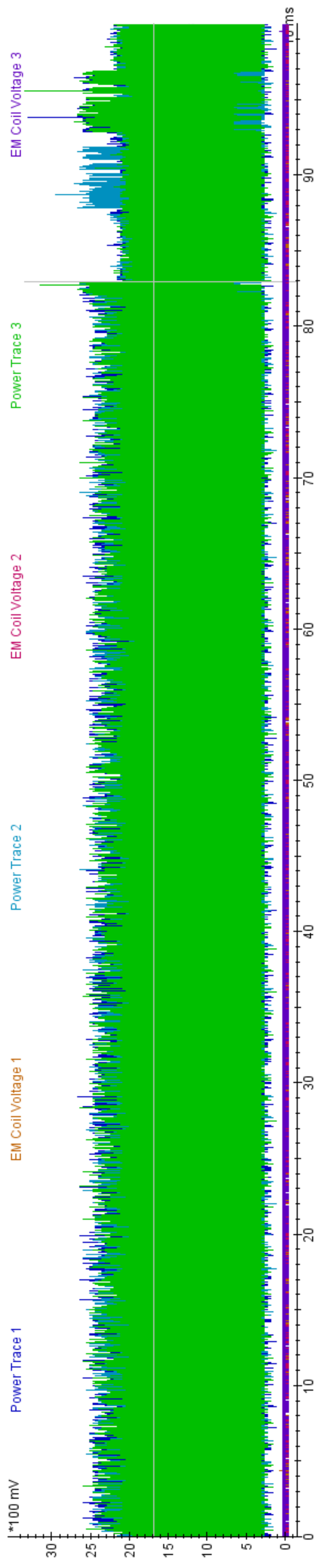


Figure D.6: 3 Power and EM Coil Voltage traces for a Type C smart card, synchronized on the Command ADPU.

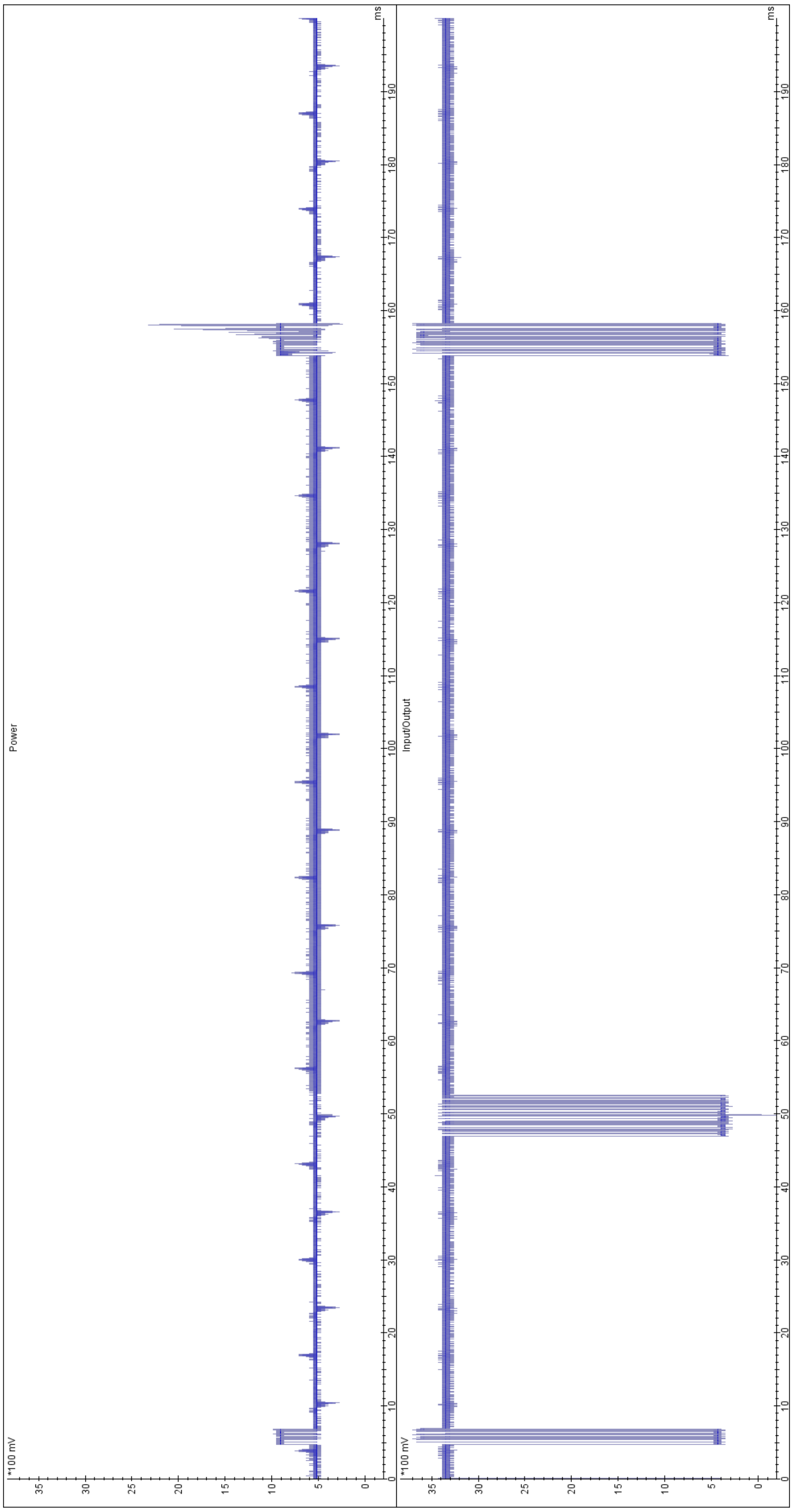


Figure D.7: A Power and I/O trace from a single execution of the function in Listing 6.2 on a Type G smart card.

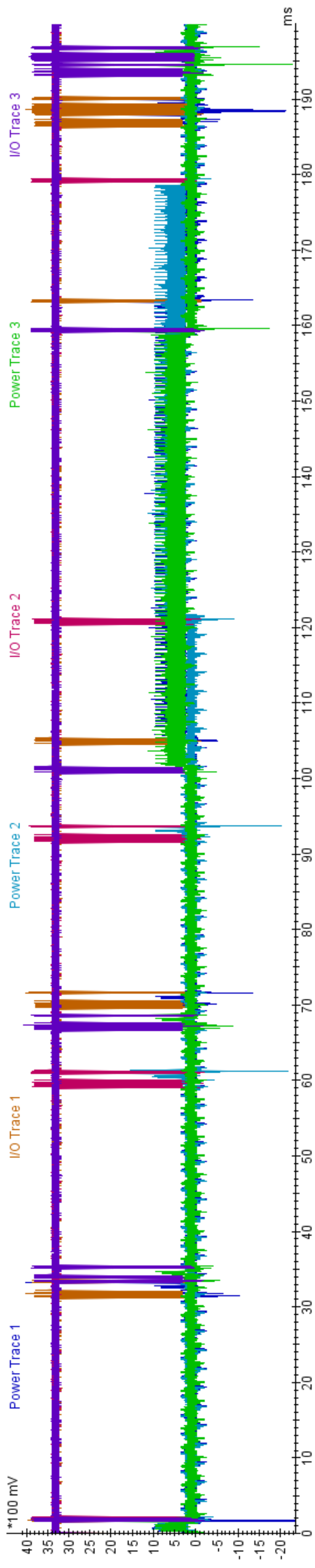


Figure D.8: A set of 3 power and I/O traces for the Type H smart card, synchronized on the Select ADPU.

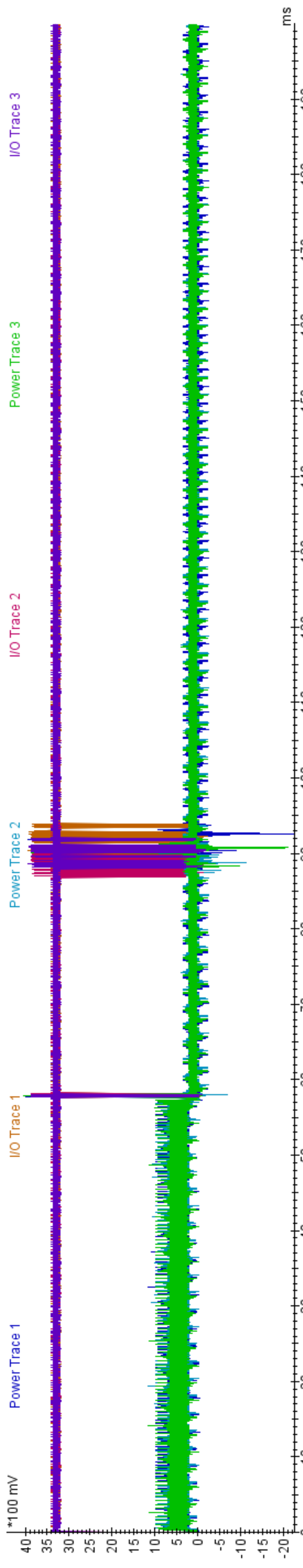


Figure D.9: A set of 3 power and I/O traces for the Type H smart card, synchronized on the Command ADPU.

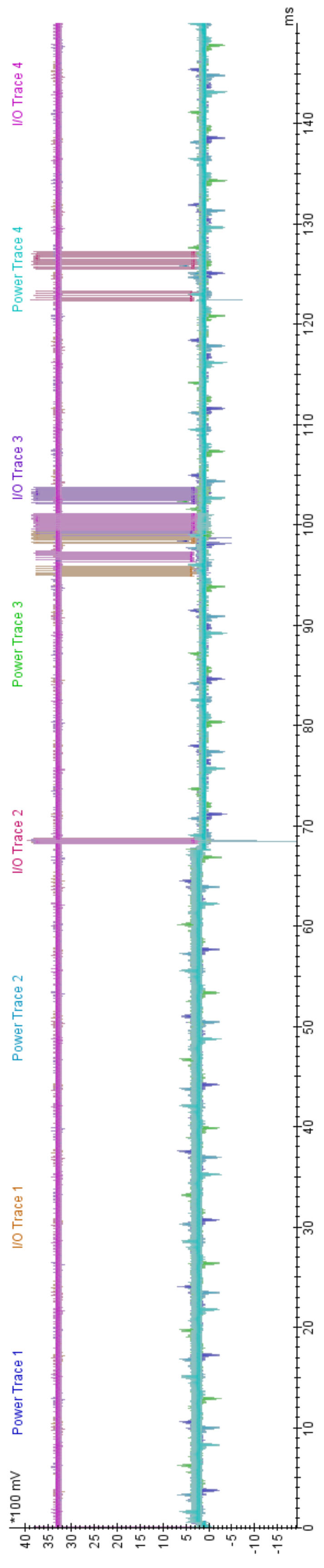


Figure D.10: A set of 4 power and I/O traces for the Type I smart card, synchronized on the Command ADPU.

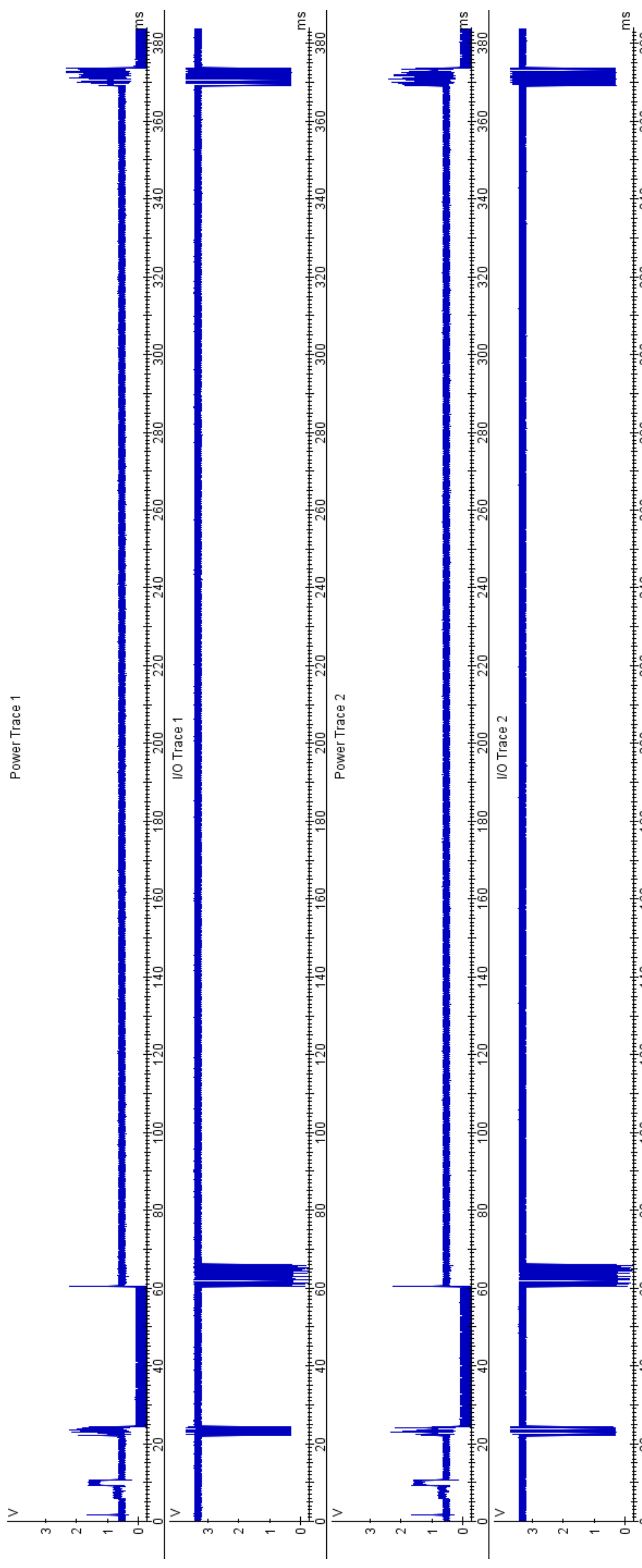


Figure D.11: A set of 2 power and I/O traces for the Type J smart card, synchronized on the Select ADPU. Note that they are practically identical.

Appendix E

Smart Card Obituaries

In the process of doing the experiments for this thesis, a total of 23 brave smart cards were put to rest. Below is an overview of how they met their untimely demise; may they rest in pieces.

Type	**Casualties**	**Means of Death**
Type A	4/5	1x Decapping, 2x Laser, 1x EM-FI - Glitched Boot ROM
Type B	1/2	1x EM-FI - Changed ATR - Muted
Type C	1/3	1x EM-FI ≈2 Million executions - Muted
Type D	0/2	0x Deaths
Type E	0/1	0x Deaths
Type F	12/15	6x Decapping, 2x laser, 3x VCC Glitching - Muted, 1x EM-FI - Muted
Type G	1/2	1x EM-FI - Changed ATR - Muted
Type H	3/4	2x VCC Glitching 3000+ executions - Muted, 1x EM-FI 3000+ executions - Muted
Type I	1/5	1x Melted - Temperature/VCC Glitching[a]
Type J	0/1	0x Deaths

[a]This was done by one of my colleagues at Riscure[9]

Table E.1: An overview of the deaths of the smart cards used during the making of this thesis.