

# Design and Validation of a Platform for Electromagnetic Fault Injection

Josep Balasch

imec-COSIC KU Leuven

Kasteelpark Arenberg 10, B-3001 Leuven, Belgium

Email: josep.balasch@esat.kuleuven.be

Daniel Arumí, Salvador Manich

Universitat Politècnica de Catalunya - BarcelonaTech

Avinguda Diagonal, 647, 08028 Barcelona, Spain

Email: daniel.arumi@upc.edu, salvador.manich@upc.edu

**Abstract**—Security is acknowledged as one of the main challenges in the design and deployment of embedded circuits. Devices need to operate on-the-field safely and correctly, even when at physical reach of potential adversaries. One of the most powerful techniques to compromise the correct functioning of a device are fault injection attacks. They enable an active adversary to trigger errors on a circuit in order to bypass security features or to gain knowledge of security-sensitive information. There are several methods to induce such errors. In this work we focus on the injection of faults through the electromagnetic (EM) channel. In particular, we document our efforts towards building a suitable platform for EM pulse injection. We design a pulse injection circuit that can provide currents over 20 A to an EM injector in order to generate abrupt variations of the EM field on the vicinity of a circuit. We validate the suitability of our platform by applying a well-know attack on an embedded 8-bit microcontroller implementing the AES block cipher. In particular, we show how to extract the AES secret cryptographic keys stored in the device by careful injection of faults during the encryption operations and simple analysis of the erroneous outputs.

## I. INTRODUCTION

The advent of the *Internet of Things* is shaping an ecosystem of ubiquitous, smart and interconnected electronic devices with promising applications and services. Yet this trend is not without risks. Built-in functionalities are necessary to enable secure communications and/or to provide strong information security guarantees. The fact that devices operate in the field make them a very attractive target for adversaries with physical access to them, who may attempt to break the security of the system driven by e.g. economical incentives.

*Fault injection* attacks are a particular class of physical attacks in which an adversary *actively* tampers with an electronic circuit and/or its environmental conditions with the aim of inducing errors. The exploitation of these errors can allow to defeat security functionalities put in place by designers. Examples are: bypassing security checks (e.g. a PIN verification routine), perturb secure hardware modules (e.g. random number generators), or extract secret cryptographic keys from the device (e.g. Bellcore attack against the RSA public-key cryptosystem [5] or DFA attacks against block cipher implementations [4]).

Methods to inject faults to electronic circuits are various and range from the classical approaches discovered by the US pay TV hacking community in the beginning of the nineties (e.g. insertion of clock glitches [8] or voltage spikes [2]), to

more advanced methods such as optical fault injection [17] or electromagnetic (EM) fault injection [14].

In this work we focus on the injection of faults through the EM channel. This technique has regained significant attention in recent years, particularly due to: i) the possibility to inject faults in a non-invasive manner, i.e. without need to perform chip decapsulation, and ii) the enabling of fine grained temporal and spatial resolutions, i.e. allowing to target local parts of a circuit.

**Related Work.** In their seminal work on EM fault injection, Quisquater and Samyde [14] used a camera flash-gun to inject high voltages into the coil of an active probe, leading to the generation of eddy currents in the surface of a chip which caused memory errors. A few years later, Schmidt and Hutter [16] employed a spark-gap transmitter to generate strong EM bursts and radiation over a microcontroller, and used this setup to break an RSA-CRT implementation. Since then, different EM fault injection setups have appeared in the literature. An overview of state-of-the-art constructions is given by Maurine [9], who differentiates between two types of platforms:

- 1) EM *harmonic* injection platforms, which use a micro-antenna to expose a circuit to continuous sinusoidal EM waves. Signals are provided by an RF generator typically followed by a power amplifier. Such a setup has been shown to disturb analogue blocks such as internal clock generators or ring-oscillator based constructions [3].
- 2) EM *pulse* injection platforms, which use often hand-made EM injectors to generate abrupt variations of the EM field on the vicinity of a circuit. Signals are provided by high-voltage pulse generators with low timing resolutions. Such setups have been tested on cryptographic modules [7], embedded microcontrollers [10], and generic, exemplary circuits [11].

**Our Contributions.** In this work we focus on the second category of platforms, namely, for EM pulse injection. Our main contributions are:

- 1) We design and construct a platform to perform fault attacks based on EM pulse injection. Instead of relying on commercially available equipment for high-voltage pulse generation, we develop our own flexible yet low-cost solution for suitable EM pulse generation.

- 2) We validate our platform by applying a well-known DFA attack against an (unprotected) AES software implementation running on an embedded 8-bit controller.

## II. A PLATFORM FOR EM PULSE INJECTION

A typical platform for EM pulse injection is made up of a pulse injection circuit (PIC), an EM injector (EMI), a stepper table (ST) and a desktop computer (PC), see Fig. 1. The device under test (DUT) is placed on the ST table and under the influence of the EMI. The DUT package may be partially or totally removed to increase the effect of the pulse. The PC can communicate with the DUT to trigger certain operations, typically a security-related function such as data encryption or signature generation.

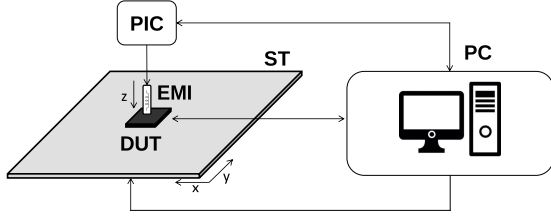


Figure 1: High-level view of a typical EM pulse injection platform.

The aim of the attack is the injection of a fault at a specific time point during the run of the security-related function. Triggering the PIC generates a high current transient in the EMI coil, producing an abrupt magnetic pulse which couples with the DUT. If injected successfully, the fault will alter the expected execution of the function and, as it will be shown later, this can then be exploited by an attacker in order to break the security of the system.

The key elements of the platform are the EMI and the PIC designs. With respect to the EMI there are several proposals in the literature (see e.g. [12]) that can be selected. In our experiments we have used a custom EMI made of a 6 loops coil wound over a cylindrical ferrite nucleus of 1.5 mm diameter. The design of the PIC module is not trivial because of its specific requirements. It must produce transients with high current during short periods of times, and triggered at a very specific time instant. Earlier works [7], [10], [11] have employed commercial high-voltage pulse generators (up to 200 V, current 8 A). Other works have used professional tools for hardware security evaluation, e.g. Riscure's Inspector FI in [15], with currents up to 100 A.

These demanding requirements are hardly met in workbenches assembled with standard laboratory equipment only, because of the limitations of the equipments themselves and the parasitic effects present in the interconnections. In what follows we present our design of a PIC module achieving similar requirements. In particular, the proposed platform is able to inject faults for currents over 20 A.

### A. Pulse Injection Circuit

The schematic of the PIC module is presented in Fig. 2 together with the power supply and control unit. The PIC

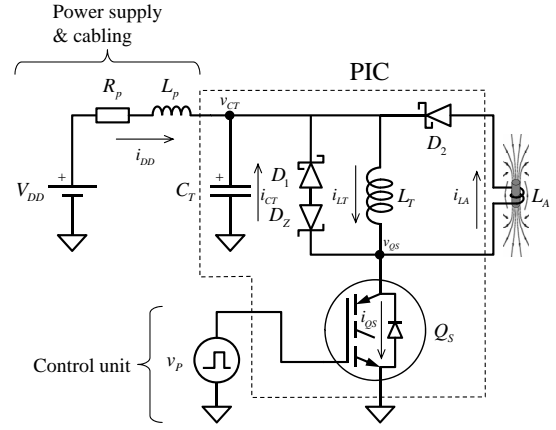


Figure 2: Simplified schematic of our pulse injection circuit.

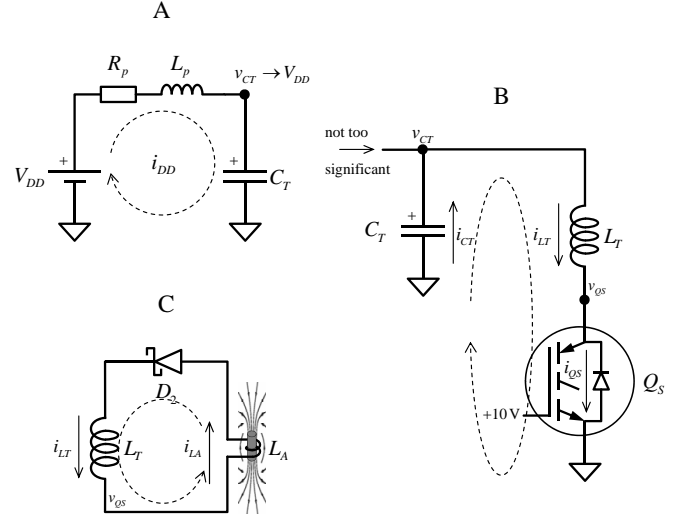


Figure 3: The three phases of the pulse injector circuit.

module itself is shown inside the dotted line area and is made of the following elements: a tank capacitor  $C_T$ , a tank coil  $L_T$ , an IGBT transistor switch  $Q_S$ , two over-voltage protection diodes  $D_1$  (Schottky) and  $D_Z$  (Zener) and a return Schottky diode  $D_2$ . The EMI component is the coil  $L_A$ .

The tank capacitor  $C_T$  is a low parasitic capacitor which in fact is made of a net of different rated capacitors connected in parallel with a total capacitance of 3.5 mF. The tank coil  $L_T$  is low parasitic and high quality factor wound in a toroidal nucleus of 200 mH. The IGBT transistor switch  $Q_S$  is a fast power transistor controlled by an insulated gate whose maximum ratings are 1.2 kV collector-to-emitter voltage, 30 A pulsed current and  $\pm 20$  V gate-to-emitter voltage. The over-voltage protection ( $D_1, D_Z$ ) is used to protect the IGBT from the induced coil over-voltage. It is rated by the Zener transistor  $D_Z$  whose break-down voltage is 1.2 kV. Finally, the return diode  $D_2$  is a Schottky type to minimize the power dissipation and it can withstand a non-repetitive peak surge current of 50 A and a reverse voltage of 20 V. All the components inside the PIC module are carefully routed and closely placed in order to

keep the parasitic effects due to cables and connectors down to a minimum. Between all the elements the most crucial ones are the tank coil  $L_T$ , the EMI component  $L_A$  and the Schottky diode  $D_2$ . The cost of all components in the PIC is around 50 euros.

Externally the PIC module is supplied by a power source  $V_{DD}$  of 20 V that is connected to the module using standard banana cables whose parasitics are modeled by  $R_p$  and  $L_p$  components. The triggering of the module is made by the pulsing voltage source  $v_P$  that switches ON/OFF  $Q_S$  with gate-to-emitter voltages of 10/0 V.

### B. Working principle

The operation of the PIC module follows a three phase cycle as illustrated in Fig. 3. During the first phase (A)  $C_T$  is charged at  $V_{DD}$ , in the second phase (B) it is discharged through  $L_T$  achieving a high current  $i_{LT}$  and finally in the third phase (C) this high current is derived through the EMI  $L_A$  and quickly discharged so that in the EMI itself the current rises and falls during a very short time interval. Thus creating the magnetic pulse in the ferrite nucleus.

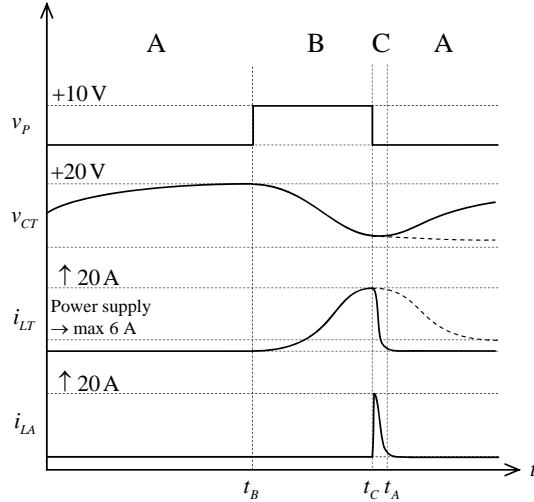


Figure 4: Voltages and currents involved in the PIC module.

In Fig. 4 the representation of the main currents and voltages is shown. During phase A,  $C_T$  is charged as formerly explained and the main currents of the circuit are 0 because  $Q_S$  is OFF. The current provided by the power source may be low because of the long time phase. Typically more than 3 ms are allowed.

To prepare the EMI shot, phase B is activated by switching ON  $Q_S$  after rising  $v_P = 10$  V. Rapidly  $C_T$  discharges through  $L_T$  and  $Q_S$  forming a current loop as illustrated in Fig. 3. These three components form a second order filter that presents an overshoot in current  $i_{LT}$  as illustrated in Fig. 4. Due to the large values and high quality factors of  $C_T$  and  $L_T$  the overshoot is significantly high, in our case larger than 20 A or one order of magnitude more than the current provided by the  $V_{DD}$ . During this current increase the contribution of  $V_{DD}$  is marginal because of the relative long time constant response of the cabling parasitic components ( $R_p, L_p$ ) and the limits

of the power source. If  $Q_S$  is kept ON for a period larger than 500  $\mu$ s, then the  $i_{LT}$  overshoot would damp reaching the  $V_{DD}$  limiting current, 1 A in our case. Similarly,  $v_{CT}$  would decrease to few volts, the cutting voltage of  $V_{DD}$ . This later case is illustrated by discontinued lines in the figure.

However, once  $i_{LT}$  reaches the maximum (time  $t_C$  in Fig. 4 before 500  $\mu$ s after  $t_B$ ) we can switch OFF  $Q_S$  by setting  $v_P = 0$ . The strong reactive behavior of  $L_T$  induces a sudden voltage spike in node  $v_{QS}$  that can easily reach 1 kV or even more, it will depend on the  $L_A$  inductance. As a result the circuit consisting of  $L_T, L_A$  and  $D_2$  is closed and the maximum current  $i_{LT}$  loops through  $L_A$  (see Fig. 3). The rising current slope in  $L_A$  is very steep as illustrated in Fig. 4, typically 400 mA ns<sup>-1</sup>. This current loop lasts in C until the magnetic energy accumulated in  $L_T$  dissipates completely, slightly more than hundred microseconds later than  $t_C$ . In Tab. I the current equations involved in each phase are listed concisely.

Table I: Currents present in each phase of the PIC.

Phase	Currents	
A	$i_{CT} = -i_{DD}$	$i_{QS} = 0$
	$i_{LT} = 0$	$i_{LA} = 0$
B	$i_{CT} \uparrow$	$i_{QS} = i_{CR}$
	$i_{LT} = i_{CR}$	$i_{LA} = 0$
C	$i_{CT} = -i_{DD}$	$i_{QS} = 0$
	$i_{LT} \downarrow$	$i_{LA} = i_{LT}$

### C. Characterization

Measurements of the electromagnetic effects induced by the EMI are made. The probe shown in Fig. 5 is used. It consists of a flat coil of 6.6 mm diameter with 7 turns that is connected to a 50  $\Omega$  resistor at which the current  $i_{probe}$  is measured. The EMI component is placed over the surface of the probe at a distance of 3.24 mm.

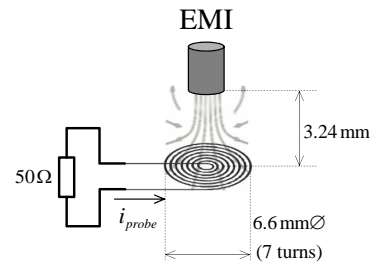


Figure 5: Current probe and its layout with the EMI component.

In Fig. 6 oscilloscope screen-shoots are presented with the currents measured in the EMI component  $i_{LA}$  and the probe  $i_{probe}$ . The right view is of a large scale and the large dissipating period of  $i_{LA}$  can be clearly appreciated. It lasts about 120  $\mu$ s. In addition the increase is seen as a vertical transition in which the transition time cannot be appreciated. At the top of this transition the probe current  $i_{probe}$  is shown like a narrow impulse. On the left view a zoom of the rise transition is made. It can be observed that  $i_{LA}$  over-shoots at 48 A after 100 ns to

immediately decrease at a steady current of 28 A. The impulse generated in the probe  $i_{probe}$  reaches a maximum of 800 mA.

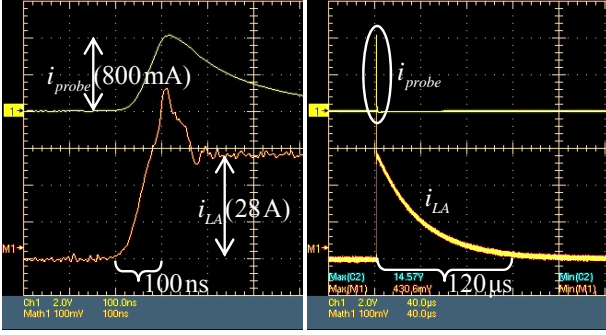


Figure 6: Oscilloscope screenshots of the EMI current  $i_{LA}$  and probe induced current  $i_{probe}$ . At the right large and at the left detailed views are presented.

Figure 7 presents the control unit voltage  $v_P$  and the currents at the tank coil  $i_{LT}$  and the power transistor  $i_{QS}$  in phase B. For comparison reasons here the transistor is not switched OFF so that the current is let to decrease at the level of the power supply  $i_{DD}$ . It can clearly be appreciated that the maximum current is 23 times larger than the power supply one. Also, time instants  $t_B$  and  $t_C$  are labeled. In normal operation after time  $t_C$  this current ( $i_{QS}$ ) would be instantly cut at zero.

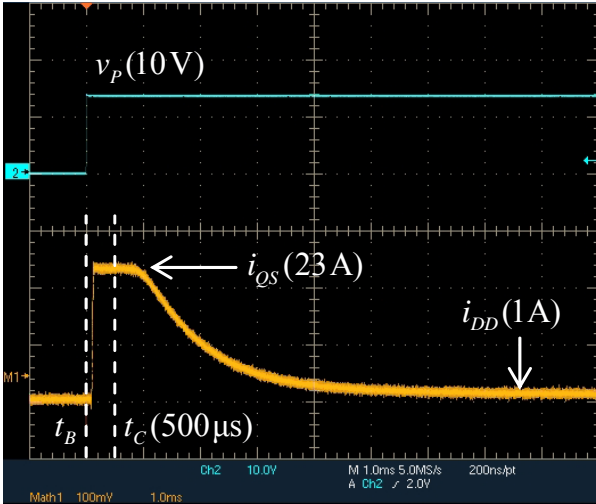


Figure 7: Oscilloscope screenshot of the power transistor current  $i_{QS} = i_{LT}$  when the transistor is kept permanently ON. Maximum current in relation to the power supply  $i_{DD}$  can be appreciated. Time instants  $t_B$  and  $t_C$  are labelled.

The PIC module presented in this section is a particular instantiation of our overall design. Therefore it can be scaled up if necessary. In particular, the present design can be replicated up to four times and connected in a star like shape configuration around the EMI component to multiply the current injection by this same factor while maintaining the same transition times. In this case, however, the EMI

component would need to be modified to accommodate the higher currents.

### III. PLATFORM VALIDATION

In this section we provide an experimental validation of our fault injection platform by performing a classical key-recovery DFA attack [4] on a cryptographic block cipher implementation.

We select as DUT an 8-bit ATmega328P microcontroller mounted on an Arduino Uno board. The device, manufactured by using a  $0.35 \mu\text{m}$  process, runs at 16 MHz. In order to maximize the effect of the EM pulse injection, we have exposed the back-side of the chip by removing the epoxy layer using mechanical grinding and milling (as described in [6]) without damaging the chip. The device is programmed with a software implementation of the Advanced Encryption Standard (AES), a symmetric encryption algorithm standardized by NIST [1]. This cipher takes as input a 128-bit input plaintext  $P$  and outputs the corresponding 128-bit ciphertext  $C = E_k(P)$  encrypted with a 128-bit secret key  $k$ , internally stored in EEPROM memory. The implementation takes around  $270 \mu\text{s}$  to execute (equivalent to 4366 cycles at 16 MHz).

Our complete experimental setup contains the elements depicted in Fig. 1. We place the Arduino Uno board on top of the ST and interface it to an external PC through a serial communication channel. The PC can generate and send arbitrary plaintexts  $P$  to the device and fetch the corresponding ciphertexts  $C$ . The voltage levels and pulse periods of the power supply and control unit driving the PIC are remotely configurable from the PC. In our experiments, we additionally use an oscilloscope to monitor the power consumption of the DUT, i.e. by measuring the voltage drop of a  $0.1\Omega$  shunt resistor in series with the circuit ground line. As explained later, we use this information to locate the temporal point in which to inject the fault. Finally, we position the EMI above the depackaged DUT. A close-up view of our setup is shown in Figure 9.

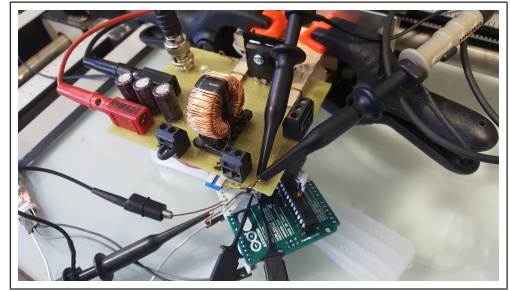


Figure 9: Close-up view of our experimental setup: PIC with EMI above DUT.

#### A. Piret's DFA on AES

Extracting secret cryptographic keys from block cipher implementations is a well-researched topic. One of the most classic attacks, which we select for our experiments, is described by Piret and Quisquater in [13] and targets implementations of

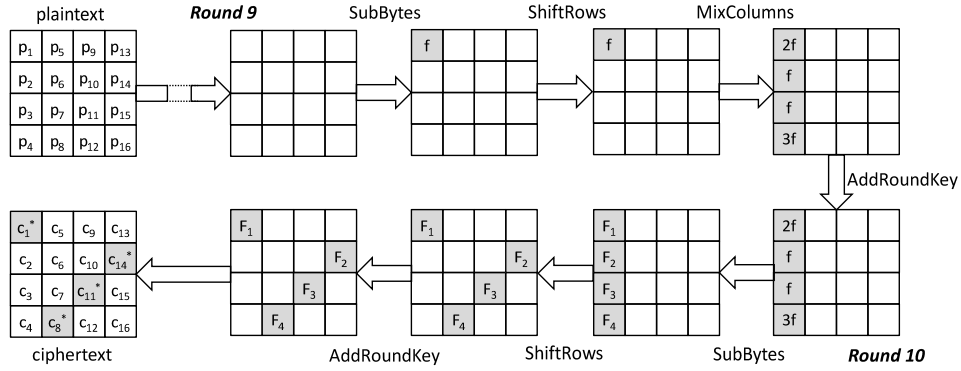


Figure 8: Output propagation of a byte-fault error at the beginning of Round 9 of AES-128.

the AES. We apply the attack on an instantiation of the AES-128, the version of AES that uses a 128-bit secret key  $k$ . The AES-128 takes as input a data block of 128 bits  $P$ , which is internally arranged as a  $4 \times 4$  matrix of bytes (the so-called *AES state*). The AES state is gradually modified by iterating over 10 transformation rounds which perform the following internal operations: *AddRoundKey* consists on a byte-wise XOR between AES state and AES round sub-key, *SubBytes* is a byte-wise non-linear operation which can be instantiated as a substitution table, *ShiftRows* performs a row-wise rearrangement of the AES state bytes, and *MixColumns* performs a linear combination of bytes within a column. Note that *MixColumns* is not applied in Round 10.

The DFA attack by Piret and Quisquater assumes an adversary capable of injecting a *random* fault into a single byte of the AES state *before* the *MixColumns* operation in Round 9. From the properties of the cipher, such an error will propagate to the ciphertext affecting 4 out of 16 bytes. This is illustrated in Figure 8, assuming a fault is injected into the first byte of the AES state. We denote the injected error by  $f$ , which is the XOR difference between the correct and erroneous value of the byte  $s_{00}^* = s_{00} \oplus f$ . The error  $f$  propagates linearly until the end of Round 9, affecting a full column of the AES state. At the beginning of Round 10, the error is transformed through the non-linear *SubBytes* operation and ultimately reaches the output ciphertext affecting bytes at positions 1, 8, 11 and 14, i.e.  $c_1^* = c_1 \oplus F_1$ ,  $c_8^* = c_8 \oplus F_4$ ,  $c_{11}^* = c_{11} \oplus F_3$  and  $c_{14}^* = c_{14} \oplus F_2$ .

Given a correct/incorrect ciphertext pair, i.e.  $(C, C^*)$ , it is then possible to construct the following set of equations using the equalities between  $f$ ,  $2f$  and  $3f$  at the end of Round 9:

$$S^{-1}(c_1 \oplus k_1) \oplus S^{-1}(c_1^* \oplus k_1) = 2 \times [S^{-1}(c_{14} \oplus k_{14}) \oplus S^{-1}(c_{14}^* \oplus k_{14})] \quad (1)$$

$$S^{-1}(c_{14} \oplus k_{14}) \oplus S^{-1}(c_{14}^* \oplus k_{14}) = S^{-1}(c_{11} \oplus k_{11}) \oplus S^{-1}(c_{11}^* \oplus k_{11}) \quad (2)$$

$$S^{-1}(c_{11} \oplus k_{11}) \oplus S^{-1}(c_{11}^* \oplus k_{11}) = 3 \times [S^{-1}(c_8 \oplus k_8) \oplus S^{-1}(c_8^* \oplus k_8)], \quad (3)$$

where  $S^{-1}$  is the inverse *SubBytes* operation and  $(k_1, k_8, k_{11}, k_{14})$  are secret key bytes used in that last execution of *AddRoundKey*. Iterating over all possible  $2^8$  values of the sub-key bytes in these equations allows to narrow down the list of possible candidates. In fact, and as explained in [13], if the adversary has 3 correct/incorrect ciphertext pairs the values  $(k_1, k_8, k_{11}, k_{14})$  can be recovered with a 100% probability. Note that the adversary does not need to know the value  $f$  of the injected fault, since the attack exploits the propagation of this error to the output ciphertext.

## B. Results

Reproducing the DFA attack in our experimental setup requires an adequate tuning of the *temporal* and *spatial* settings of our fault injection platform. Namely, we need to ensure that i) the EM pulse is generated at the exact time when the device performs operations at the beginning of Round 9, and ii) the EMI is positioned at a suitable spot above the surface of the DUT that results in computation errors.

We first set the temporal occurrence of the fault by using the information provided by the power consumption waveform of the circuit during an AES execution. An example is shown in Fig. 10 (bottom). The 10 transformation rounds (highlighted by the vertical dotted lines) yield very similar patterns in the power consumption waveform, which allows us to identify the time instant where Round 9 begins. By setting delays to our control unit we can synchronize the instant where  $v_P = 0$ , i.e. where phase (C) is activated, to the beginning of computations in Round 9. This is shown in Fig. 10 (top). We then set the spatial resolution by placing the EMI at different positions over the depackaged DUT using the ST. Determining whether the position enables the attack is done experimentally. We select a random plaintext  $P$ , collect its corresponding ciphertext  $C$  (without faults), and then we run our fault injection setup at different positions until we obtain an erroneous ciphertext  $C^*$ .

Quite naturally, not all injected faults correspond to the fault model assumed in Piret's attack. So we need to perform some fine-tuning trials in order to find a suitable temporal and spatial location that results in an error in exactly four ciphertext bytes. These should correspond to  $(c_1^*, c_8^*, c_{11}^*, c_{14}^*)$ , i.e. same as depicted in Fig. 8. In our experimental setup, the desired



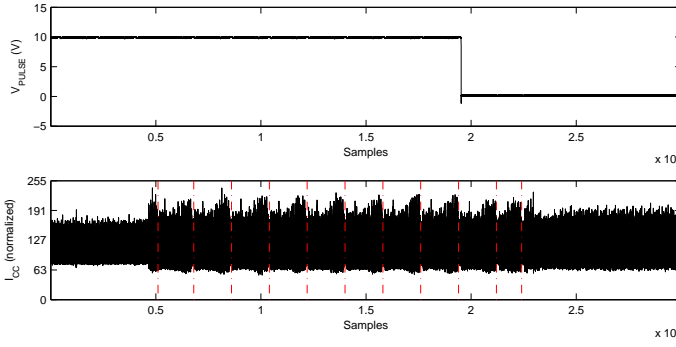


Figure 10: Oscilloscope captures during AES execution: value of  $v_P$  (top), normalized value of  $i_{CC}$  (bottom). Vertical lines show the different rounds of AES.

faults can be obtained when placing the EMI on a corner above the decapsulated DUT, indicating that this part of the chip layout contains some components that are susceptible to the injected EM pulse. With this configuration, we can obtain 3 suitable  $(C, C^*)$  pairs that allow to recover  $(k_1, k_8, k_{11}, k_{14})$ . The remaining secret key bytes are simply obtained by repeating the attack to generate errors on different ciphertext bytes. This can be done by shifting the temporal occurrence of  $v_P = 0$  until we observe errors at positions  $(c_2^*, c_5^*, c_{12}^*, c_{15}^*)$ ,  $(c_3^*, c_6^*, c_9^*, c_{16}^*)$  and  $(c_4^*, c_7^*, c_{10}^*, c_{13}^*)$ , respectively.

### C. Discussion

In order to wrap-up our analysis, we provide here a brief explanation of the actual effect of the injected fault to the DUT. The SubBytes transformation is typically implemented in software by means of a look-up table called *Sbox*. Since the *Sbox* table is an algorithm constant, it is instantiated directly in non-volatile memory. The implementation of SubBytes is thus a simple iteration over all state bytes performing the substitution  $s_i = Sbox(s_i)$  for  $1 \leq i \leq 16$ . In our DUT, the instruction LPM is the one used to load elements from an array in program memory to a register. We have identified this operation as the one that is affected by the EM pulse injection. The error caused by the fault is simple: the instruction fails to load the desired value and, instead, it stores into the register the *previous* value on the bus. Put differently: given two consecutive instructions  $s_{i-1} = Sbox(s_{i-1})$  and  $s_i = Sbox(s_i)$ , a fault injected on the latter operation will cause  $s_i = Sbox(s_{i-1})$ . In practice, and due to the properties of the algorithm, this effect resembles a random fault model as assumed in [13], therefore enabling the attack. Note also that the effect of the fault is restricted to this particular operation, i.e. we observe no alterations in the subsequent instructions.

## IV. CONCLUSION

Fault injection attacks are amongst the most threatening techniques that an adversary can use to undermine the security of embedded devices. In this work we have described our efforts to build a suitable and low-cost platform for EM fault injection, which can be used by designers for evaluation purposes. Our

design can be scaled-up to accommodate stronger requirements and improved by selecting more directional EMI elements. We plan to investigate these aspects in future works.

### ACKNOWLEDGMENT

This work was supported in part by the Research Council KU Leuven: C16/15/058, Cathedral ERC Advanced Grant 695305, and by Spanish TEC2013-J41209-P government project. The authors would like to thank Arthur Beckers for providing the decapsulated ATmega controller used in the experiments.

### REFERENCES

- [1] Specification for the Advanced Encryption Standard (AES). Federal Information Processing Standards (FIPS) Publication 197, 2001.
- [2] H. Bar-El, H. Choukri, D. Naccache, Michael Tunstall, and C. Whelan. The sorcerer's apprentice guide to fault attacks. *Proceedings of the IEEE*, 94(2):370–382, Feb 2006.
- [3] 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 *COSADE 2012*, volume 7275 of *LNCS*, pages 151–166. Springer, 2012.
- [4] Eli Biham and Adi Shamir. Differential fault analysis of secret key cryptosystems. In Burton S. Kaliski Jr., editor, *Advances in Cryptology - CRYPTO '97*, volume 1294 of *LNCS*, pages 513–525. Springer, 1997.
- [5] Dan Boneh, Richard A. DeMillo, and Richard J. Lipton. On the importance of checking cryptographic protocols for faults (extended abstract). In *EUROCRYPT '97*, volume 1233 of *LNCS*, pages 37–51. Springer, 1997.
- [6] Jakub Breier, Chien-Ning Chen, Wei He, Alexander Herrmann, and Marc Stöttinger. Low Cost Laser Fault Target. Available at: <http://iacr.org/workshops/ches/ches2014/presentations/rump/02-Chen.pdf>, 2014.
- [7] Amine Dehbaoui, Jean-Max Dutertre, Bruno Robisson, and Assia Tria. Electromagnetic transient faults injection on a hardware and a software implementations of AES. In *FDTC 2012*, pages 7–15. IEEE Computer Society, 2012.
- [8] Oliver Kömmerling and Markus G. Kuhn. Design principles for Tamper-Resistant Smartcard Processors. In *USENIX Workshop on Smartcard Technology*, pages 9–20. USENIX Association, 1999.
- [9] Philippe Maurine. Techniques for EM fault injection: Equipments and experimental results. In *FDTC 2012*, pages 3–4. IEEE Computer Society, 2012.
- [10] Nicolas Moro, Amine Dehbaoui, Karine Heydemann, Bruno Robisson, and Emmanuelle Encrenaz. Electromagnetic fault injection: Towards a fault model on a 32-bit microcontroller. In Wieland Fischer and Jörn-Marc Schmidt, editors, *Fault Diagnosis and Tolerance in Cryptography - FDTC 2013*, pages 77–88. IEEE Computer Society, 2013.
- [11] Sébastien Ordas, Ludovic Guillaume-Sage, and Philippe Maurine. EM injection: Fault model and locality. In Naofumi Homma and Victor Lomné, editors, *Fault Diagnosis and Tolerance in Cryptography - FDTC 2015*, pages 3–13. IEEE Computer Society, 2015.
- [12] Sébastien Ordas, Ludovic Guillaume-Sage, Karim Tobich, Jean-Max Dutertre, and Philippe Maurine. Evidence of a larger em-induced fault model. In *CARDIS 2014*, volume 8968 of *LNCS*, pages 245–259. Springer, 2014.
- [13] Gilles Piret and Jean-Jacques Quisquater. A differential fault attack technique against SPN structures, with application to the AES and KHAZAD. In Colin D. Walter, Çetin Kaya Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2003*, volume 2779 of *LNCS*, pages 77–88. Springer, 2003.
- [14] Jean-Jacques Quisquater and David Samyde. Eddy current for Magnetic Analysis with Active Sensor. In *Esmart 2002*, 2002.
- [15] Jasper van Woudenberg Rajesh Velegali, Robert Van Spyk. Electro magnetic fault injection in practice. In *International Cryptographic Module Conference - ICMC 2013*, page 6, 2013.
- [16] Jörn-Marc Schmidt and Michael Hutter. Optical and EM Fault-Attacks on CRT-based RSA: Concrete Results. In *Austrochip 2007*, pages 61 – 67. Verlag der Technischen Universität Graz, 2007.
- [17] Sergei P. Skorobogatov and Ross J. Anderson. Optical fault induction attacks. In *CHES 2002*, volume 2523 of *LNCS*, pages 2–12. Springer, 2002.