A Versatile Framework for Implementation Attacks on Cryptographic RFIDs and Embedded Devices

Timo Kasper, David Oswald, Christof Paar

Horst Görtz Institute for IT Security, Ruhr-University Bochum, Germany
{Timo.Kasper,David.Oswald,Christof.Paar}@rub.de

Abstract. We present a unified framework for advanced implementation attacks that allows for conducting automated side-channel analysis and fault injection targeting all kinds of embedded cryptographic devices including RFIDs. Our proposed low-cost setup consists of modular functional units that can be interchanged, depending on the demands of a concrete attack scenario. We give details of customized modules for the communication with many types of embedded devices and other modules that allow to inject various types of faults. An FPGA-based approach enables very accurate timing and flexible adaption to any extension module. The corresponding data acquisition system for side-channel attacks makes precise power and EM analyses possible. Our setup facilitates the promising combination of active and passive techniques, which is known to render many established security countermeasures ineffective. We introduce several methods for the automatic profiling of cryptographic devices and model their behaviour both with respect to side-channel analysis and fault injection. To demonstrate the capabilities of our framework, we perform the first practical full key-recovery on a cryptographic contactless smartcard employing Triple-DES reported in the literature and inject multiple faults in a widespread microcontroller. We thereby disprove the common belief that highly sophisticated and expensive equipment is required to conduct such attacks. Rather, we illustrate a cost-effective setup that can be tailored to any desired type of security evaluation or penetration test.

Keywords: Side-channel, fault injection, security evaluation, models, RFID, mobile and embedded computing

1 Introduction

There exist solutions for both symmetric and asymmetric cryptography that are highly secure from the mathematical point of view. It is well-known that, when these cryptographic mechanisms are realised in practice, unprotected implementations are vulnerable to *passive attacks*, i.e., power analysis (Differential Power Analysis (DPA), Simple Power Analysis (SPA), template attacks), and *active*

attacks, i.e., fault injection or microprobing. This especially matters in the case of embedded devices, such as Radio Frequency Identification Devices (RFIDs), smartcards, remote controls and mobile computing devices, which potential attackers can obtain in large numbers.

Countermeasures exist for both types of attacks, however, it is not clear how susceptible the secured devices are to a combination of power analysis with various fault-injection techniques. Furthermore, many manufacturers do not seem to care or are not aware of the attacks, hence many unprotected (or badly protected) cryptographic devices are currently used in security-sensitive applications in the field.

There is a lack of realistic, practically verified models of the adversary, especially in the case of fault injections. Each cryptographic device can show a significantly different behaviour, that is, an RFID device is vulnerable to different types of attacks than the microcontroller of a smartcard or an Application Specific Integrated Circuit (ASIC). Hence it is necessary to investigate the susceptibility of each new target, e.g., which types of faults can be injected with which success rate or whether multiple fault injections are realistic. For each device, an in-depth profiling is necessary, which accordingly has to be automated. We focus on those attacks that are realistic for an adversary with a limited budget (typical university lab equipment) and that rely on public domain or publicly available, self-made solutions for the equipment, where possible.

The general structure of our versatile framework enabling active and passive implementation attacks, as well as their combination, is presented in Sect. 2. One part of our setup are modules as detailed in Sect. 3 that provide the means to communicate with any device, including wireless interfaces. Extensions allowing for the injection of various types of faults are covered in Sect. 4, while Sect. 5 is dedicated to the acquisition of information leakage emanated by the device under test. In Sect. 6 we exemplify the comprehensive capabilities of our framework by practically analysing the security of two widespread commercial products, i.e., a cryptographic contactless smartcard and a microcontroller. We demonstrate a full key-recovery of the secret key of the Triple-DES hardware employed in the former RFID device by means of side-channel analysis, and an automatic profiling with respect to faults of the latter, revealing the parameters for a subsequent, practically verified injection of multiple faults with a success rate of almost 100%. We aim to show that penetration tests, tampering with cryptographic devices and complex side-channel analyses do not require costly tools and equipment as used in the labs of the industry, but can rather be performed with inexpensive or self-built equipment.

1.1 Classification of Implementation Attacks

Implementation attacks are well documented in the literature [30, 28, 27], hence we do not give a detailed compendium of the possible implementation attacks here, but rather classify the attacks and specify the scope of our framework. Figure 1 highlights the focus of this article, i.e., DPA, SPA, fault injection and their combination. We do not cover invasive attacks here which rely on directly

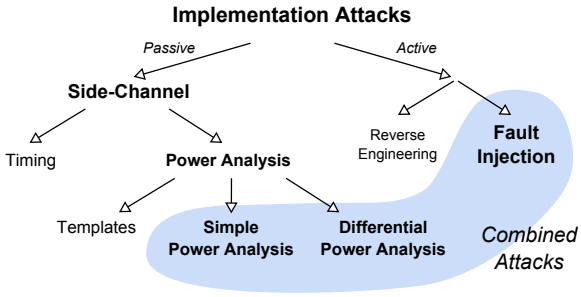

Fig. 1. Classification of implementation attacks.

tampering with the silicon wafer, e.g., probing attacks [16] or reverse engineering by taking microscopic photos of all layers of a chip [33]. For state-of-the-art implementations of cryptographic algorithms, these attacks generally demand for highly sophisticated equipment and require a very strong and well-funded adversary, while we are considering an ordinary attacker and low-cost equipment.

2 System Overview

The here proposed system is designed on a flexible modular basis such that it can be adapted to test the security of any cryptographic device. The different functional units are categorised in modules for the communication with the DUT, modules for side-channel analysis, and modules for fault injection. These modules may in turn consist of a set of smaller sub-modules that are detailed in the following subsections, and that allow for arbitrary extensions of the framework according to the requirements of the evaluator. Note that their tasks may partially overlap, e.g., parallel and serial communication can be carried out directly from the controlling Personal Computer (PC), from a microcontroller of any submodule, or by means of the Field Programmable Gate Array (FPGA) mainly used for fault injection. Likewise, processing of the data and digital filtering is not restricted to software inside the PC but can also be realised in hardware on the FPGA.

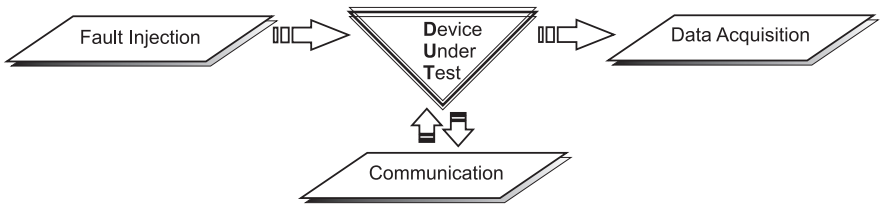

Fig. 2. Modules for side-channel analysis of a cryptographic device.

As a Device Under Test (DUT) we take into account any microcontroller, FPGA, ASIC, or other embedded system including contactless devices and RFIDs. In case of specialities, e.g., concerning communication via Radio Frequency (RF), or if special faults are desired, it is straightforward to incorporate additional modules to the ones described in this article. Figure 2 illustrates the structure of the functional units as detailed in the following.

For carrying out side-channel attacks, generally an adversary must have access to the plain- or ciphertext that is processed by the DUT, in order to evaluate measurements or determine whether a fault was successful. This information is in most cases delivered by the communication modules described in Sect. 3.

For passive side-channel attacks, the behaviour of the DUT with respect to timing, power consumption, Electro-Magnetic (EM) emanation, etc. has to be accurately monitored. The corresponding data acquisition module described in Sect. 5 serves for this purpose, while the recorded data is often post-processed and evaluated by a controlling PC.

The fault injection module detailed in Sect. 4 takes care of the active aspects of side-channel analysis. Due to the variety of faults that can be injected to the DUT, this FPGA-based functional group is designed most versatile, such that it can be extended to induce literally any type of fault.

3 Communication Modules

Though it is possible to carry out most side-channel attacks using commercially available equipment, customised hardware for communicating with the DUT is highly advantageous. Commercial readers often rely on proprietary Integrated Circuits (ICs) that carry out certain tasks automatically, e.g., a built-in Random Number Generator (RNG) will generate the nonces for a challenge-response-protocol, compute the correct parity bits and checksums, data will be encrypted, encoded, and sent, and this all happens without that an adversary can directly influence the process. Thus, in the following we present customised readers for the relevant standards that are tailored to the requirements of implementation attacks, in order to gain complete control over the communication, i.e., send arbitrary bits, send repeatedly the same chosen plaintext, intentionally compute wrong checksums and — most important in the context of side-channel attacks — have complete control over the timing and generate reliable trigger signals. Note that it is often sufficient to implement only some part of the protocol, until the DUT performs the targeted cryptographic operation, e.g., encryption, which happens often at the beginning of the communication.

For many practical attacks, additionally a device is required that can serve as a replacement for the original DUT, e.g., an emulated smartcard that can be fully controlled by the adversary. For each form of communication covered in this article, we describe such an emulation extension that is compatible with the corresponding reader. Accordingly, in combination with the self-built reader devices, communication data can be monitored and unknown protocols can be automatically reverse-engineered.

3.1 Communication with RFIDs

A minimum RFID system consists of two main components, namely a reader generating a sinusoidal magnetic field which supplies the second component of the system often called tag, transponder or contactless smartcard, with energy and often a clock. Both components are equipped with a coupling element, e.g., a coil, that allows for data transfer in both directions. In the literature, a reader is sometimes named Proximity Coupling Device (PCD), and a contactless smartcard called Proximity Integrated Circuit Card (PICC). In the context of wireless devices, attacks based on measuring the EM field, e.g., a Differential Electro-Magnetic Analysis (DEMA) [9, 23], are obviously most convenient, since the whole circuitry is packaged, e.g., in plastic cards, and hence neither a contact interface nor the chip itself is accessible to an attacker in a non-invasive attack scenario. Previous results [2, 15, 36] suggest that this approach is suitable for a wide range of RFID devices.

The following section we detail communication modules for contactless smartcards according to the ISO 14443 standard [20], operating at a frequency of $f_c = 13.56\,\text{MHz}$, that are widely deployed in various security sensitive applications such as the electronic passport to store biometric data, RFID-enabled credit cards, and access control systems. Contactless smartcards have sufficient energy to perform complex computations and are hence capable of using both symmetric and asymmetric state-of-the-art cryptography, e.g., an Elliptic Curve Cryptography (ECC) engine in the electronic passport (ePass) is used to verify signatures, and 3DES or AES is often used to encrypt the current balance in the context of contactless payment systems.

A similar module has been developed for communicating with RFIDs operating on 125 kHz (as presented in Sect. 3.1), which are mainly used for car immobilizers and access control. Compared to contactless smartcards, these devices possess less computational power, thus often simple (and often insecure) proprietary ciphers are used here, if cryptography is used at all.

Contactless Smartcards For the communication with contactless smartcards, we employ a self-built embedded system [22] consisting of a multi-purpose reader that is based on a freely programmable Atmel ATMega32 [6] microcontroller, an ISO 14443 compliant RF interface and some components for signal processing. Various types of antennas and amplifiers can be connected, e.g., for increasing the activation- or eavesdropping range. A second device that cooperates with the reader is designed to appear like an authentic tag to an RFID reader, i.e., can emulate any contactless smartcard, and furthermore can acquire the information contained in the field. Both devices allow for a comprehensive control of the communication on the physical layer, i.e., every single bit sent and received as well as the RF field is completely controlled by the adversary with an accurate timing of $\approx 75\,\text{ns}$. Communication with a controlling PC takes place via an USB interface, and reliable trigger signals can be issued at any instant during the protocol. Further details as well as all schematics and layouts to build the here

employed devices for contactless smartcards at a cost of less than 40 $ are fully made public in [21].

While for side-channel analysis of RFIDs as practically detailed in Sect. 6.1 mainly the reader functionality is used, it is also possible to practically perform other attacks, e.g., a simple replay attack or a relay attack in the field [22]. The software running on the microcontroller has been vastly improved, such that active relay attacks can be carried out, i.e., the information transmitted can be modified in real-time. This is amongst others useful to falsify the Unique Identifier (UID) of an RFID tag, that is usually fixed in the hardware, or to enforce communication at a lower data rate than the original one.

We have fully implemented the authentication protocols for several contactless smartcards employing 3DES, AES and proprietary ciphers. The precise timing control of the RF field has amongst others been advantageous in the context of spoofing the random number generator built into Mifare Classic cards [34]. By exactly fixing the timing when the RF field is switched on and when the commands are sent during the authentication, the attacked Mifare card will always generate the same fixed value instead of random numbers, which extremely facilitates key-recovery attacks and allows to reveal a full 48 Bit key of a Mifare Classic card — much more efficiently than all previously reported attacks — in seconds [24].

125 kHz RFID Tags The module for communicating with RFID tags operating in the range of 100 kHz to 150 kHz is designed similar to the ISO 14443 module. Again, it is controlled by an ATMega32 microcontroller, while an Atmel U2270B IC (price: approx. 1 $), which is capable of all relevant modulation schemes and a typical data rate of 5 kBaud, takes care of the handling of the RF communication. Due to the small data rate we opted for a standard RS232C serial communication with the controlling PC, instead of an USB port. The schematic is similar to the one given in Application 3 in the datasheet [7].

3.2 Communication with Contact-based Smartcards

For the side-channel analysis of contact-based smartcards according to ISO 7816 [1] we have built an adaptor with the appropriate dimensions and the specified contact interface on the one side, which fits into any commercial smartcard reader. On the other side of the adaptor, a socket for smartcards allows any ISO 7816 card to be plugged in. The device allows for relay attacks with contact-based smartcards, and facilitates implementation attacks on smartcards. The data and power wires are tapped and rewired, such that the bitstreams can be relayed from and to a standard reader, e.g., for the analysis of communication protocols. The power lines allow to connect an external stable power supply (or our module detailed in Sect. 4.6 for inducing power faults), while a variable resistor is inserted in series with the ground pin of the smartcard for performing power analyses. Similar to the tools for contactless cards and RFIDs a smartcard can be emulated, a relay attack can be conducted or only the pure reader functionality can be used, while simultaneously faults can be injected or measurements for

side-channel analysis can be recorded. We have successfully tested the adaptor in combination with our measurement setup by performing a power-analysis of an 8-Bit smartcard by Atmel containing an AES implementation in software: The correct 128-Bit key was revealed in minutes from approx. 100 measurements.

3.3 Arbitrary Parallel/Serial Communication

Embedded systems and cryptographic devices that do not feature an RFID- or ISO 7816-based interface usually employ a serial or parallel protocol to communicate with their environment. In the context of implementation attacks, example targets may be FPGAs that are configured with an encrypted bitstream or cryptographically protected USB dongles. Therefore, we support a variety of corresponding protocols, either by the controlling PC if the timing is not crucial (e.g., USB, RS-232, or parallel port) or by means of the fault injection FPGA platform, if precise timing is required (e.g., Serial Peripheral Interface (SPI) and general purpose I/O pins). If necessary in future applications, further methods can easily be added thanks to the modular nature of our setup.

4 Fault Injection Modules

Many different approaches can be utilised to inject faults in ICs. In order to unify the application of this methods, we propose an FPGA-based control board which is extended with *fault modules* that realise the actual physical effect. The FPGA provides an RS-232 interface to the controlling PC, supervises the injection of faults with precisely adjustable parameters (e.g., position in time, duration etc.) and is able to communicate with the DUT if required.

4.1 Modelling Fault Injection

Before detailing the diverse methods to inject faults in ICs, we identify general properties of faults in order to provide a model that helps to characterise the requirements for concrete attacks.

Permanence If a fault injection permanently alters the DUT, for instance, destroys a hardware part or overwrites the firmware, it is said to be *permanent*. Otherwise, if the fault only affects the outcome of a limited number of computations, it is *non-permanent* or *transient*.

Precision of Time Position Subsequent attacks may require the fault to occur either at a *random* (indeterminate) position, within some *region* or at a *precisely determined* point in time.

Number of Affected Bits A fault is called *single-bit* fault if it alters exactly one bit, or *multi-bit*, if it changes ≥ 2 bit, e.g., the state of a complete register.

Effect The induced modification can manifest itself in a *bit flip*, i.e., logic values are inverted, a *fixed state*, i.e., logic values are tied to 0 or 1, or *inconsistent behaviour* of the DUT. In the latter case, the fault injection causes inconsistencies in the state of a device by affecting a distinct part of its control

logic. A common example of this effect is the skipping of instructions on a microcontroller, e.g., due to the instruction pointer being incremented but the current instruction not being executed.

Despite the need for theoretical models, we would like to stress here that in practice it is often difficult, even in case of a successful fault injection, to exactly determine which part of the device is malfunctioning due to the fault, i.e., what exactly happened in the internal circuits of the attacked cryptographic device when it shows a certain behaviour. This applies particularly for black box analyses, where an attacker knows nothing about the implementation.

4.2 Types of Physical Faults in Integrated Circuits

There is a variety of ways to trigger faulty behaviour of ICs, differing (amongst others) in complexity, cost, effectiveness and the possible effects caused by the fault. In the following, we give a brief survey of methods that have been proposed in the literature.

Microprobing One of the most direct yet complicated fault injection methods is to de-package the silicon die and contact a specific circuit path using *microprobes*. As detailed in [26], the attacker is able to exactly monitor the waveforms present on the tapped wire, or can actively modify the value, for instance by short-circuiting it to ground. Due to the immediate access to the DUT, virtually all types of faults can be injected. Moreover, the method allows for reverse engineering of the circuit. However, the needed equipment is expensive (in [26], the authors estimate a cost of 10 000 - 100 000 $) and requires considerable skill and experience to be handled efficiently. Additionally, the invasive nature of the attack makes it unusable in scenarios where permanent, obvious physical modification of the DUT is not desired. For these reasons, we do not consider microprobing attacks in this article.

Temperature Variation Since the characteristics of circuit elements vary with temperature, ICs only work correctly within the temperature range specified by the vendor. Thus, cooling or heating the DUT and operating it outside of its maximum specifications can lead to faulty behaviour. High or low temperature especially affects memory cells and can cause random modification of Static Random Access Memory (SRAM) cells or disable read/write operations of Non-volatile Memory (NVM), i.e., Electrically Erasable Programmable Read-Only Memory (EEPROM) or Flash [14]. Generally, exact timing of the fault is complicated due to the limited thermal conductivity of the IC package and the die itself. Besides, most of the fault parameters mentioned in Sect. 4.1 are hard to control with this approach, limiting the possible application scenarios. In the current version of our setup, temperature variations can only be applied manually, i.e., using coolant spray or heating devices.

Optical Effects By exposing the circuit to white or laser light, electron-hole pairs are created that can cause current flow at p-n junctions [38, 14] of semiconductors, resulting in changes of logic levels in the affected region of the IC, e.g., switch a transistor. By applying a mask to focus a small area, optical faults allow for precise targeting of certain parts of a circuit, down to the single-transistor level [42], with fine control over the fault effect. Note that inducing optical faults is a semi-invasive attack, as the plastic packaging of the chip has to be opened mechanically or by etching, which is straightforward for standard IC packages, e.g., Dual Inline Package (DIP) or Small-Outline Integrated Circuit (SOIC), but can become infeasible for an adversary in the case of sophisticated smartcards.

Variation of Power Supply Temporarily increasing (*positive glitch*) or reducing (*negative glitch*) the supply voltage of an IC to a certain level is a well-established method to inject faults [13, 12], particularly with regard to the skipping or misinterpretation of processor instructions. As the power is supplied via an external pin (for the case of most embedded device) or the surrounding EM field (for contactless (RFID) devices), the fault injection path is easily accessible, allowing for non-invasive attacks. However, at the same time, this single entry point can also be disadvantegous from an attacker's point of view: Countermeasures such as monitoring or filtering the supply voltage before it enters the core of the circuit are relatively inexpensive, because they only need to be implemented for one isolated section of the IC.

Electro-Magnetic Pulses Transients of the EM field cause induction of currents in conductors and can thereby change logic levels present on an IC. In contrast to power glitches, the fault injection is not performed over a single wire. Rather, the fault can affect any part of the DUT, making it harder to prevent and detect than variations of the supply voltage. This approach is especially suited for RFIDs [17], for which direct access to the power supply would require an invasive manipulation of the antenna connection.

Variation of an External Clock For devices with external oscillators, i.e., for which manipulations of the clock signal are feasible, slightly modifying the clock period for one or few (half-)cycles may lead to data corruption [26]. Due to different delays of distinct circuit paths, values that take longer to propagate (e.g., because they are transported over the *critical path*[1]) may not be handled correctly in the following clock cycle.

4.3 FPGA-based Platform for Fault Injection

The use of an FPGA has certain advantages compared to a microcontroller-based solution, particularly with regard to precise timing of control signals at high

[1] The critical path is the register-to-register path with the largest delay and thus limits the maximum clock frequency

clock frequencies. To minimize the design time, we use the commercial Xilinx Spartan-3 board [44] as a basis. The system clock frequency is set to 100 MHz, so that the FPGA runs significantly faster than most of the considered embedded systems, which are usually clocked at between $\approx$ 32 kHz and 20 MHz [11], thus enabling the injection of faults at multiple instants during one clock cycle of the DUT. Note that at higher frequencies, it becomes increasingly difficult to select and apply the involved analogue (and digital) components appropriately. To simplify the implementation of complex control logic, our design is built around a general 8-bit microcontroller softcore (Xilinx PicoBlaze, cf. [45], available as VHDL source file for Xilinx FPGAs) which is internally connected to several application-specific modules, as depicted in Fig. 3.

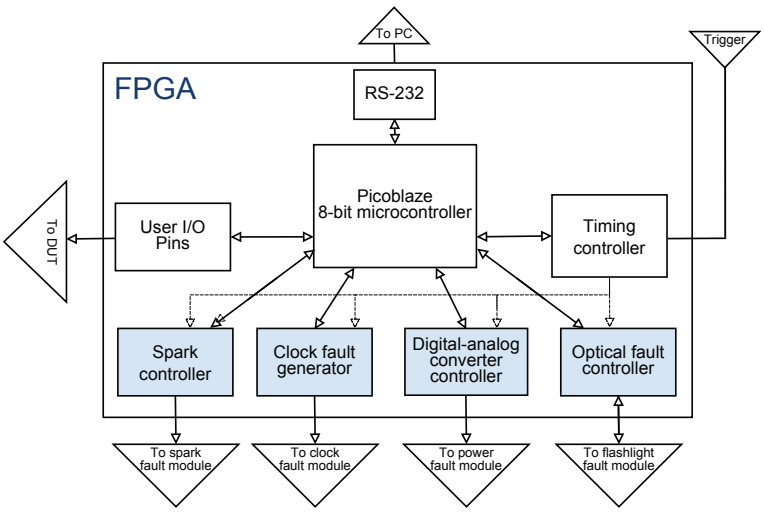

Fig. 3. Internal structure of the control FPGA for fault injection.

The PicoBlaze softcore is a Reduced Instruction Set Computing (RISC) microcontroller programmable using a simple assembler language [43]. It has low resource requirements (96 slices + 1 block Random Access Memory (RAM) on a Spartan 3 FPGA) and is well suited for implementing non timing-critical control and interface logic.

All timing-critical operations that have to respond to external inputs instantly, and require guaranteed timing behaviour, are moved into the application specific blocks. The central module in this respect is the *timing controller*, which is responsible for starting previously configured faults with precise timing parameters. The purpose of the microcontroller is to provide a unified and extensible interface for the controlling PC to setup the timing controller and the fault injection modules. Currently, we support fault injection by means of optical methods, EM pulses, power glitches and variations of the clock signal.

4.4 Optical Fault Injection

A modified electronic flash of a photo-camera serves as the basis for a low-cost module for optical fault injections as described, e.g., in [38]. A small Printed Circuit Board (PCB) that is connected to the flash of the camera could be re-used for our purposes with only small modifications. It mainly consists of a High Voltage Generator (HVG) that produces up to 400 V DC out of the 3 V battery supply of the camera, and a 220 pF capacitor C_1 that is charged with electrons by the HVG. Figure 4 illustrates the principle of the module.

A pin of the PCB ('Ready') that was used to drive a green Light Emitting Diode (LED) to indicate when C_1 is fully charged, i.e., a flash is ready to be triggered, is connected to an input pin of an FPGA. The switch S_1 that had been mechanically switched when pressing the button of the camera, more precisely, when the lens opens to take a picture, was replaced by a transistor and can now also be controlled by the FPGA. Turning it on instantly discharges C_1 into the flash and thereby releases an optical fault. The FPGA is programmed such that it busy-waits until the flash is ready, then starts the interaction with the DUT, and finally triggers an optical fault at the desired instant.

If a coil is connected instead of the flash, it is reported [10] that the resulting strong magnetic field can inject permanent faults into RFIDs such as electronic passports, i.e., the device can be forever deactivated.

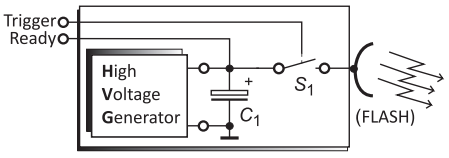

Fig. 4. Optical fault injection module.

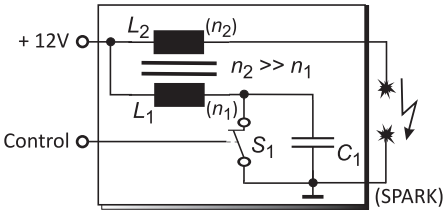

Fig. 5. Injecting faults with sparks.

4.5 Electro-Magnetic Fault Injection with Sparks

It is well-known in electronics that a sudden change in electric current generates an EM field (cf. Sect. 4.2) – the higher the amplitude and the faster the alteration, the stronger will the resulting EM field be. Thus, in the following we describe our module for the injection of faults by generating sparks, as illustrated in Fig. 5. The idea for this module is borrowed from an ignition system for petrol engines and allows for triggering a spark by means of an ignition coil, a switch S_1 realised by a high voltage Insulated Gate Bipolar Transistor (IGBT) [41], and a common spark plug for cars.

The ignition coil consists of two inductances L_1 and L_2 with different numbers of turns n_1 and n_2, that are coupled to form an electric transformer. The voltages

V_1 and V_2 over the coils L_1 and L_2 follow the equation $\frac{V_1}{V_2} = \frac{n_1}{n_2}$. The turn ratio is such that $n_2 \gg n_1$, hence any voltage occurring on the side of L_1 will be amplified orders of magnitude higher on the side of L_2. To ignite an arc, a large over-voltage pulse is required: 250 V to 300 V on the side of L_1, corresponding to approx. 20 kV on the side of L_2, are sufficient to generate a spark [31]. An STP10NK50Z IGBT employed as the switch S_1 withstands high voltages up to 500 V and can be turned on with a gate voltage of approx. 3 V. This allows for diretly switching the 'Control' signal and thereby triggering the switch S_1 in Fig. 5 by means of the controlling FPGA.

The generation of a spark consists of two phases, controlled by the time during which S_1 is switched on: During the first phase, the coil needs to be charged for a minimum amount of time. Then, in the second phase, the spark is released by opening the switch. While the switch S_1 is turned on, a DC current flows through the coil L_1, and hence charges L_1 with energy, until S_1 opens the connection: The sudden interruption of the electric current flow through L_1 implies that the magnetic field collapses rapidly, inducing a high voltage on the side of L_1, whose amplitude depends on how much energy has been stored during the charging phase. The much higher voltage transformed to L_2 instantly ignites the desired spark at the spark plug, while the capacitor C_1 limits the voltage overshooting to protect the switch from getting damaged.

The strength of the induced fault can be steered by varying the amount of time during which the ignition coil can store energy. During our tests, a charge phase of 5 ms was sufficient to produce reliable, strong sparks. As a power source we are using a 12 V car battery, as it supplies very large currents. If a standard power supply is used, it is recommended to connect a very large capacitor ($> 10\,000\,\mu$F) in parallel to the power source, to provide a sufficient amount of current. Optionally, a coil could be connected instead of the spark plug to generate a purely magnetic field.

Note that extra caution has to be taken when conducting this type of fault injection in order to prevent destruction of the DUT or other nearby equipment[2]. Hence for every unknown device tests should be carried out starting with a big distance between the arc and the DUT. Note that, in the near field the field strength falls as a function of the distance r to the DUT, i.e., proportional to $\frac{1}{r^3}$, hence small changes in r have a strong effect on the outcome of the faults.

4.6 Power Fault Injection

As described in Sect. 4.2, power faults can be both triggered by positive (i.e., increase of the supply voltage) and negative (i.e., reduction of the supply voltage) glitches. For maximum flexibility in this respect and for fine control over the actual waveform, we have chosen a Digital-Analogue Converter (DAC) based approach, as depicted in Fig. 6.

[2] In most countries it is illegal to use the here described module, because the EM emanation can cause radio waves that disturb other electronic equipment, and the device is thus not compatible to FCC rules. It is recommended to perform tests with EM faults in a shielded environment (aluminum foil is usually sufficient).

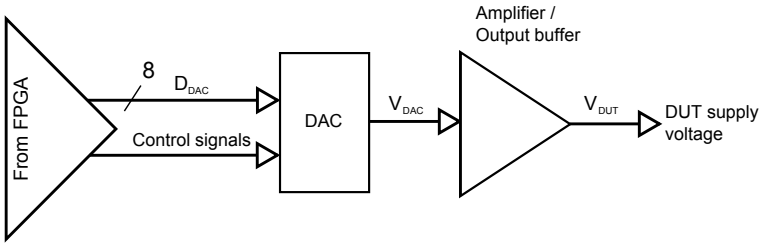

Fig. 6. Principle of the module for generating power faults.

The voltage V_{DAC} at the DAC output pin can be controlled via an 8 bit bus, passing a binary-encoded number $D_{DAC} \in \{0, \ldots, 255\}$. V_{DAC} is then given as $V_{DAC} = \frac{D_{DAC}}{255} \cdot V_{DAC,max}$ where $V_{DAC,max}$ denotes the maximum output voltage. Because the DAC generates a voltage of max. $\approx 1\,\mathrm{V}$ and its output current is limited to 20 mA, an additional output amplifier is required to provide higher voltages and greater driver strength.

Implementation Details A PCB has been designed with the structure introduced above. The used DAC is the AD9708 manufactured by Analog Devices [5], capable of running at max. 125 MSamples/s. The signal from the DAC is amplified up to a maximum of $\approx 5.5\,\mathrm{V}$, using the AD8058 Operational amplifier (OP) by Analog Devices [4], providing a theoretical bandwidth of 325 MHz at a gain of $+1$ and a slew rate[3] of 1000 V/μs.

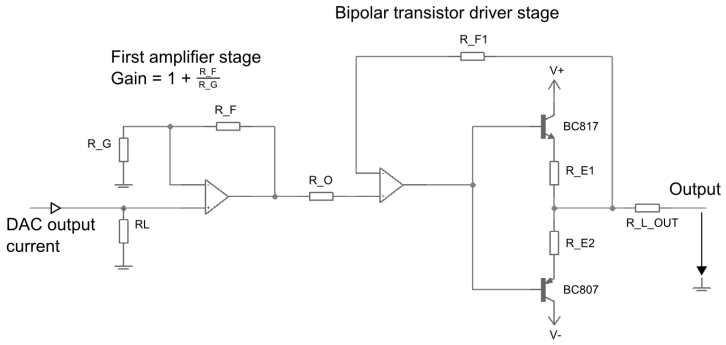

Fig. 7. Output stage of the power fault module.

[3] The slew rate indicates the maximum rate of change of the output voltage.

The OP is applied in a non-inverting configuration with the gain set to ≈ 4.7. Additionally, a bipolar transistor-based output stage according to [29] has been implemented, which enables output currents of max. 150 mA at a supply voltage of 7.5 V. By selecting different transistors, this value could be further improved if required. The schematic of the output stage is shown in Fig. 7.

As an example for the output waveform, Fig. 8 and 9 depict a 10 ns full-scale pulse generated with the proposed power fault module, recorded after the amplifier and the transistor output stage, respectively. All depicted signals have

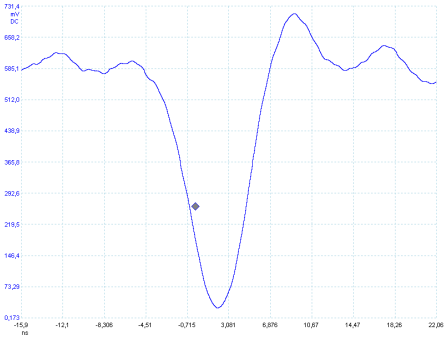

Fig. 8. Full-scale 10 ns negative voltage glitch at amplifier output, ×10 probe.

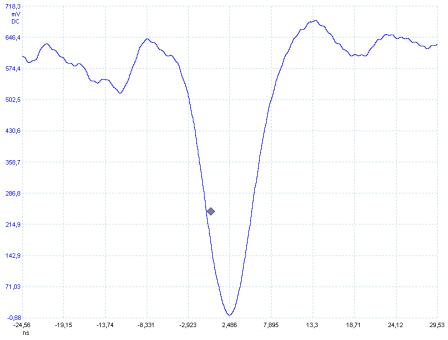

Fig. 9. Full-scale 10 ns negative voltage glitch after transistor output stage, ×10 probe.

been recorded using a probe set to ×10 attenuation to minimize the influence of the probe capacitance on the rise and fall times.

4.7 Fault Injection with Clock Variations

Clock faults are small, temporary variations of the fraction of time the clock signal is high in one period which is commonly referred to as the *duty cycle*. For maximum flexibility, a module for this type of fault has to provide

- a wide range of output frequencies, especially covering the range of embedded systems, and
- precise control over the duty cylce of the clock signal.

Our approach makes use of the Digital Clock Manager (DCM) of the Xilinx FPGA which is able to generate a clock signal with very fine control over its phase shift. By outputting both an unshifted and a shifted clock and combining these signals logically with external circuitry, several useful waveforms can be created. The module provides the signals $o_1 = clk \wedge clk_s$ and $o_2 = clk \vee clk_s$ for both shortened and stretched clock cycles. These are illustrated in Fig. 10 and Fig. 11, in which clk denotes a clock signal, clk_s this signal shifted by Δt, $\wedge$ a logical AND and $\vee$ a logical OR.

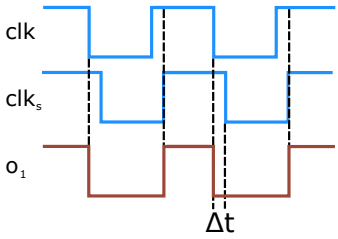

Fig. 10. Clock signal $o_1 = clk \wedge clk_s$ width shortened clock cycles.

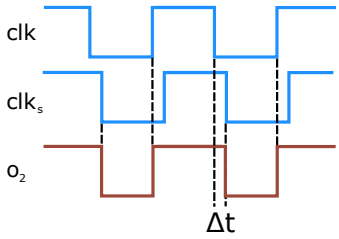

Fig. 11. Clock signal $o_2 = clk \vee clk_s$ witch stretched clock cycles.

Implementation Details The generation of the shifted clock signals clk_s is performed by the FPGA using a combination of a fine phase shift, followed by clock (down-)scaling and a coarse phase shift. The phase shift function of the Xilinx DCM is used to shift the clock by $\frac{1}{256}$th of its period. The input to the DCM is the global system clock clk_{sys} and the output is termed $clk_{sys,\,s}$ (shifted by Δ_{fine}) and both are running at 100 MHz.

clk_{sys} and $clk_{sys,\,s}$ are then passed to a prescaler (and coarse phase shifter), which toggles the output clock when an internal counter reaches half of the configured prescaling factor. The coarse phase shift by Δ_{coarse} is accomplished by pre-loading the internal counter on startup. This way, clk_s can be shifted in multiples of the system clock period, i.e., in steps of 10 ns, with respect to clk. The downscaled clocks clk and clk_s are routed via the output pins of the FPGA to the actual fault module PCB. Figure 12 summarises the complete process.

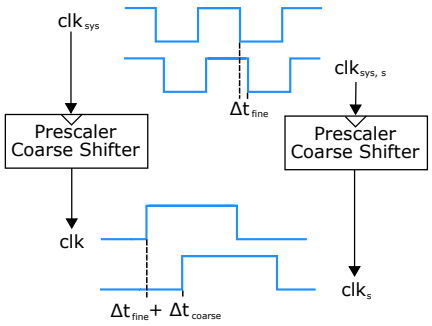

Fig. 12. Clock signal shifting and prescaling on FPGA.

On the external board, the logical operations proposed above are performed by discrete, high speed Complementary Metal Oxide Semiconductor (CMOS) ICs. According to the datasheets [37], these ICs can be operated at frequencies above 1 GHz, enabling precise adjustment of the signal timing. An example out-

put signal for o_1 (i.e., a slightly shortened clock cycle) is depicted in Fig. 13, with the prescaler set such that the output clock frequency is 16.67 MHz.

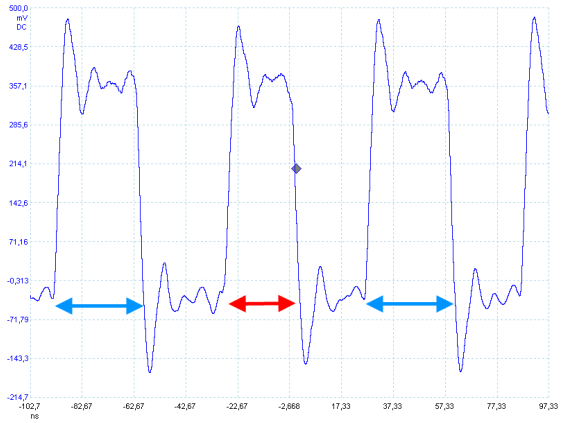

Fig. 13. Fault on 16.67 MHz clock signal, x10 probe, 20 ns/time division.

5 Data Acquisition

A controlling PC and a USB oscilloscope form the basis for the data acquisition system. The acquired data can be side-channel information (e.g., current, voltage, EM emanation or timing information), or communication data such as bitstreams in any format which then later can be evaluated by a PC. The software framework follows our modular approach and allows for straightforward substitution of its parts, e.g., when switching to a new oscilloscope or analysing a DUT that has different requirements with regard to the data necessary for mounting an attack.

In the context of combined active and passive attacks, the configuration of the fault injection device and the recording of the side-channel information can both be performed by the controlling PC, simplifying the synchronisation of the respective processes. Additionally, the PC can process the data directly after recording, so that adaptive attacks are possible, in which, e.g., a challenge is selected based on the outcome of prior steps, such as a successful fault injection.

6 Practical Attacks

In this section, we present results of attacks on real-world devices. By the example of a commercial RFID smartcard, we show the capabilities of out framework with respect to side-channel analysis. Aside, we describe an active fault injection on a widespread microcontroller.

6.1 Power-Analysing a 3DES Cryptographic Contactless Smartcard

Before detailing the analysis of an RFID smartcard employing Triple-DES, we briefly outline the applied techniques used to attack the device and propose appropriate leakage models. For the analysis, we use the well-established method of Correlation Power Analysis (CPA) [8]. As mentioned in Sect. 3.1, for RFIDs it is the natural choice to mount a non-invasive attack by measuring the EM emanation, i.e., perform a DEMA.

Modelling the Power Consumption of RFID Devices For a simple model of the frequencies where we would expect the EM leakage to occur, consider a band-limited power consumption $p(t)$ that directly affects the amplitude of the $\omega_0 = 2\pi \cdot 13.56\,\mathrm{MHz}$ frequency of the reader, i.e., the amplitude of the field will be slightly smaller in an instant when the chip requires more energy than in an instant when no energy is consumed. This results in possibly detectable frequency components in the side bands of the carrier, as depicted in Fig. 14. Equation 1 describes this model more precisely, where $\circ\!\!-\!\!\bullet$ denotes the Fourier transform[4].

$$p(t)\cos(\omega_0 t) \circ\!\!-\!\!\bullet X(j\omega) = \frac{1}{2}\left(P(j\omega - j\omega_0) + P(j\omega + j\omega_0)\right) \tag{1}$$

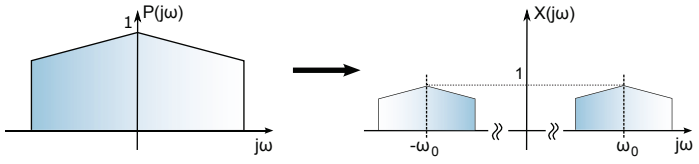

Fig. 14. Frequency spectrum of the carrier signal at ω_0 and the assumed information leakage for remote power analysis.

We refer to this approach as *Remote Power Analysis* according to Oren and Shamir [35], because the fluctuations in the power consumption of the device are modulated onto the strong carrier signal of the PCD.

Another model assumes that the internal switching of transistors on gate-level can be detected by means of near-field probes, such that the bits of a secret key might be extracted from signals in frequency bands that are independent of the carrier of the PCD. The signal model in Equation 2 for this case is additive, so that the best possible suppression of all other components - in particular the carrier frequency - is desirable. As illustrated in Fig. 15, now $P(j\omega)$ is assumed

[4] The Fourier transform is commonly used to transform signals from the time domain into the frequency domain

to be band-limited with a center frequency ω_s that is independent of the carrier frequency ω_0.

$$p(t) + \cos(\omega_0 t) \circ\!\!-\!\!\bullet\, X(j\omega) = P(j\omega) + \frac{1}{2}\left(\delta(j\omega - j\omega_0) + \delta(j\omega + j\omega_0)\right) \qquad (2)$$

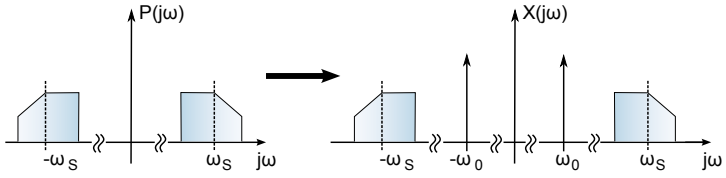

Fig. 15. Frequency spectrum of the carrier signal at ω_0 and the assumed information leakage for the additive model.

Difference-of-Means Test To determine the quality of the traces acquired with a particular measurement setup and to find out if and in which instants the traces contain data-dependent information, a Difference-of-Means (DOM) test can be conducted as a preparation for the key-recovery. The idea is to send two (or more) challenges alternately that cause a different electromagnetic emanation according to the assumed power model, acquire the corresponding traces $\boldsymbol{t}^{(1)}$ and $\boldsymbol{t}^{(2)}$ of length K, and then form four equally sized sets S_i with $|S_i| = N$ the number of traces per set.

Let $S_1 = \left\{\boldsymbol{t}_0^{(1)}, \ldots, \boldsymbol{t}_{N-1}^{(1)}\right\}$ contain the traces of the first challenge and $S_2 = \left\{\boldsymbol{t}_0^{(2)}, \ldots, \boldsymbol{t}_{N-1}^{(2)}\right\}$ those for the second challenge. The average traces $\bar{t}^{(1)}(k)$, $\bar{t}^{(2)}(k)$ and their DOM timeseries $\delta^{(1-2)}(k)$, where $0 \leq k < K$ denotes the current sampling point, are given as:

$$\bar{t}^{(1)}(k) = \frac{1}{N}\sum_{n=0}^{N-1} \boldsymbol{t}_n^{(1)}(k)$$

$$\bar{t}^{(2)}(k) = \frac{1}{N}\sum_{n=0}^{N-1} \boldsymbol{t}_n^{(2)}(k)$$

$$\delta^{(1-2)}(k) = \bar{t}^{(1)}(k) \;-\; \bar{t}^{(2)}(k)$$

In order to estimate the amount of noise, let two sets S_3 and S_4 contain uniformly distributed traces belonging to *either* challenge, i.e., assign all traces

randomly to the two sets, and define accordingly:

$$\bar{t}^{(3)}(k) = \frac{1}{N}\sum_{n=0}^{N-1} \boldsymbol{t}_n^{(3)}(k)$$
$$\bar{t}^{(4)}(k) = \frac{1}{N}\sum_{n=0}^{N-1} \boldsymbol{t}_n^{(4)}(k)$$
$$\delta^{(3-4)}(k) = \bar{t}^{(3)}(k) \;-\; \bar{t}^{(4)}(k)$$

The DOM $\delta^{(3-4)}(k)$ would ideally vanish, if the measurements contained no noise. Accordingly, maxima of $\delta^{(1-2)}$ correspond to the points in time where a data-dependent behaviour occurs — the higher the ratio between the amplitude of the peaks and the noise level, the more information is contained in the measurements, hence for more traces, a higher signal-to-noise ratio is expected. In contrast, the maxima of $\delta^{(3-4)}$, indicating the amount of noise in the traces, should all have a similar amplitude that becomes smaller with more traces acquired and would become zero for an infinite amount of traces.

Contactless Smartcard Attack In this section, we turn towards a more complex scenario and analyse a commercially available contactless smartcard. This time, we are facing a *black-box situation*, i.e., we do not know anything about the implementation of the cipher, existent countermeasures etc., so extensive profiling is necessary in preparation for a key-recovery attack. The following results base on the analysis performed in [23], which we summarise before presenting our new achievements.

The DUT is an ISO 14443-compliant RFID device (cf. [18, 19]), operating at 13.56 MHz. It features a challenge-response authentication protocol which relies on a 3DES using the two 56 bit halves of $k_C = k_1||k_2$ as the symmetric key in Encrypt-Decrypt-Encrypt (EDE) mode. The process of performing the analysis can be split up into the following steps, which we will cover in this section:

1. Align the traces in time.
2. Profile the device and locate the 3DES encryption.
3. Optimise the position of the EM probe.
4. Perform the EM analysis of the 3DES encryption.

Challenge-Response Authentication Protocol Using the RFID reader detailed in Sect. 3.1 we reverse-engineered and implemented the whole authentication protocol, but focus on the step relevant for our analyses as depicted in Fig. 16, where $\mathrm{3DES}_{k_C}(\cdot) = \mathrm{DES}_{k_1}\left(\mathrm{DES}_{k_2}^{-1}\left(\mathrm{DES}_{k_1}(\cdot)\right)\right)$ denotes a 3DES encryption involving the key $k_C = k_1||k_2$. The values B_1 and B_2 are encrypted by the PICC during the mutual authentication. B_2 originates from a random number previously generated by the PICC and is always encrypted by the PICC in order to check the authenticity of the PCD. The protocol will abort after the encryption

of B_2, in case its verification is not successful. B_1, a random value chosen by the PCD that serves for authenticating the PICC to the PCD, is mentioned here for completeness only and is not required in the context of our analyses.

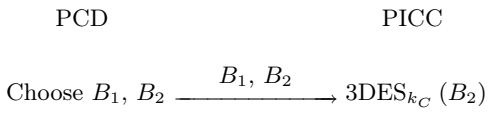

Fig. 16. Excerpt of the authentication protocol relevant for a DEMA attack.

We observe that the card unconditionally encrypts any value B_2 sent to it, hence we can freely choose the plaintext. For the CPA described in the following, we will send random, uniformly distributed plaintexts for B_2 and attack the first DES round.

Trace Preprocessing and Alignment The raw traces recorded between the last bit of the command sent by the reader and the first bit of the answer of the card do not expose any distinctive pattern, hence, digital preprocessing is applied in order to identify interesting patterns useful for a precise alignment of the traces. On the basis of the remote power model introduced in Sect. 6.1, we assume that the power consumption of the smartcard modulates the amplitude of the carrier wave at frequencies much lower than the 13.56 MHz carrier frequency, which is justified by a preliminary spectral analysis and the well-known fact that the on-chip components (such as capacitances, resistors, inductances) typically imply a strong low-pass filter characteristic.

In order to obtain the relevant side-channel information, we record raw (not demodulated) traces and perform the demodulation digitally on a standard PC, using a straightforward incoherent demodulation approach (Fig. 17, following [40]). The raw trace is first rectified, then low-pass filtered using an appropriate digital filter. An additional high-pass filter removes the constant amplitude offset resulting from the demodulation principle and low-frequency noise. Good values for the filter cutoff frequencies $f_{lowpass}$ and $f_{highpass}$ were determined experimentally and are given in this section.

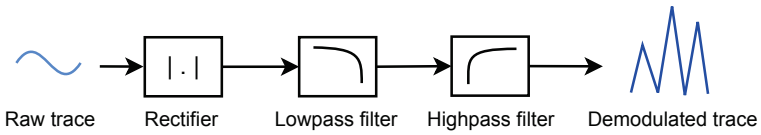

Fig. 17. Block diagram of incoherent digital amplitude demodulator.

For precise alignment during the digital processing, we select a short reference pattern in a demodulated *reference trace*. This pattern is then located in all subsequent traces by finding the shift that minimises the squared difference between the reference and the trace to align, i.e., we apply a least-squares approach. In our experiments we found that the analysed smartcard performs the operations in an asynchronous manner, i.e., the alignment may be wrong in portions not belonging to the region the reference pattern is taken from. A re-alignment has thus to be performed with respect to the part of the trace we aim to examine by means of CPA.

Probe Positioning The DOM test turns out to be an appropriate utility for finding the optimal position of the near-field probe used to capture the EM emanation. For that purpose, we implement a *Live DOM* application on the controlling PC that sends the alternating challenges to the card, records and digitally preprocesses the EM trace. After the automatic detection of the alignment pattern[5], all subsequent traces are aligned accordingly and passed to the DOM algorithm. The DOM results are displayed instantly and are continously updated while the probe can be moved. Figure 18 depicts a screenshot of the utility, where the upper window displays the processed and aligned traces with the alignment pattern highlighted, while the lower shows the squared DOM of S_1 and S_2, i.e., $\left(\delta^{(1-2)}(k)\right)^2$.

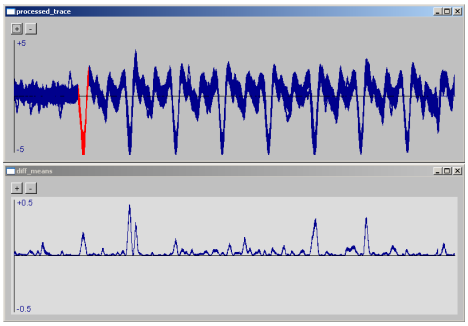

Fig. 18. Screenshot of Live DOM test utility.

By iteratively adjusting the probe position to maximise the DOM, we discovered that the results are optimal if the probe is placed directly above the IC and in parallel to the long side of the card, at a vertical angle of $\approx 35\,^\circ$ (with respect to the card plane), as shown in Fig. 19. It furthermore turned out that placing the smartcard at a vertical distance of ≈ 5 mm to the antenna considerably improves the amount of side-channel leakage.

[5] The pattern detection is accomplished by finding the first peak whose amplitude exceeds a fixed threshold

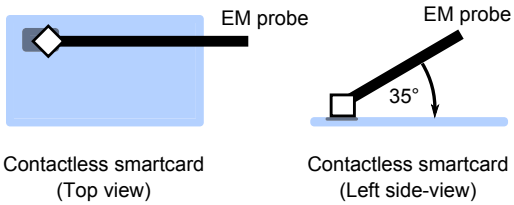

Fig. 19. Optimal positioning of EM probe.

Device Profiling As the plaintext for the targeted 3DES operation is known and the ciphertext can be computed in a known-key scenario, we are able to isolate the location of the 3DES encryption by correlating on these values. From the profiling phase with a known key it turns out that the smartcard uses an 8 bit data bus to transfer plain- and ciphertexts. The corresponding values can be clearly identified from 2000 - 5000 traces using a Hamming weight model. Figure 20 was compiled from these profiling observations, with the shape of the 3DES operation marked. The first 3DES encryption (labeled 3DES 1) results from a prior protocol step, the correlation with the correct ciphertext appears after the execution of the second 3DES (labeled 3DES 2).

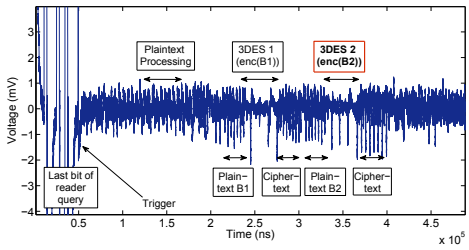

Fig. 20. Overview of operations in amplitude-demodulated trace.

3DES Engine Figure 21 shows the targeted 3DES operation identified during the profiling phase, filtered with $f_{lowpass} = 8\,\text{MHz}$ and $f_{highpass} = 50\,\text{kHz}$. The short duration of the encryption suggests that the 3DES is implemented in a special, separate hardware module, hence we assume a Hamming distance model. We also considered a Hamming weight model, but did not reach conclusive results with it.

The three marked peaks seemingly appear at the end of one complete Single-DES and are thus promising candidates as alignment patterns. We conduct a CPA on demodulated traces aligned to each of these peaks, where we consider the Hamming distance between the DES registers (L_0, R_0) and (L_1, R_1), i.e, the

state before and after the first round of the first Single-DES. When performing a standard CPA with 1 000 000 traces, correlation peaks with maximum amplitude occur for the correct subkey candidate of each S-Box, at a position which we consider as the start point of the first DES. The results are given in Fig. 22.

We observe that 1. the correlation for the output of some S-Boxes is significantly stronger than for others (e.g., for S-Box 1 and 3, for which the correct subkey can already be identified after 150 000 traces), 2. several peaks appear at different points in time for one S-Box and 3. the point of maximum correlation varies depending on the S-Box.

As the attack works (albeit after a large number of traces), we suppose that no masking scheme [30] is used to protect the hardware engine. Rather, we conjecture that hiding in the time dimension is used, i.e., dummy cycles with no computation or similar measures, to prevent correct alignment of the traces. This assumption is justified by the above observation that more than one peak occurs in the correlation curve and further strengthened by the fact that even when repeatedly sending the same plaintext B_2 to the smartcard, the shape of the DES operation and the position of the peaks depicted in Fig. 21 vary[6].

Note that with our measurement setup, recording one million traces takes approx. two days, i.e., we achieve a rate of approx. 700 measurements per minute. Having extensively profiled the DUT, we are able to focus on the relevant region of the EM trace and thus achieve substantial savings both with regard to disk space and processing time. Therefore, our attack is still feasible in a practical scenario, despite the considerable amount of traces.

[6] This misalignment also hinders improving the SNR by means of averaging.

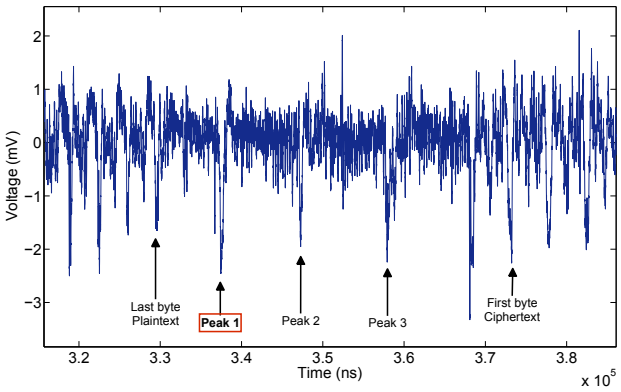

Fig. 21. Part of a trace with 3DES encryption, filtered with $f_{lowpass} = 8\,\text{MHz}$, $f_{highpass} = 50\,\text{kHz}$.

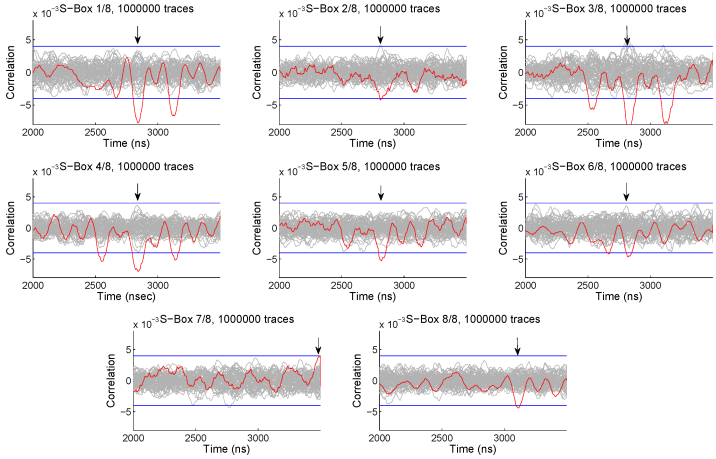

Fig. 22. Correlation coefficients for CPA after 1 000 000 traces, $f_{lowpass} = 8\,\text{MHz}$, $f_{highpass} = 50\,\text{kHz}$.

6.2 Fault Injection Attacks

After demonstrating the capabilities of the developed framework with regard to passive side-channel analysis, we address active fault injection and accordingly carry out an attack against a widespread 8-bit microcontroller, the PIC16F687 [32]. Note that the main focus in the following is on the demonstration of the fault injection capabilities, not on the implementation of actual attacks against cryptographical algorithms.

Single Faults Consequently, we start with defining a simple test scenario, attempting to skip one instruction executed by the microcontroller. The microcontroller executes a simple program to detect that a fault has been successfully injected. After the initialization, a status pin `PIN_STATUS` (connected to an FPGA input) is constantly set to high in a first infinite loop. In a subsequent infinite loop, the same pin is pulled low and additionally, another status pin `PIN_STATUS_2` is repeatedly toggled, thereby indicating whether the microcontroller is still alive.

Our attack targets the `goto` instruction at the end of the first loop. Without external influence, the DUT never exits this loop. The aim of the fault attack is to jump over this instruction, so that the second loop gets executed. This condition can be detected by checking for `PIN_STATUS = 0`, indicating a successful fault injection. To provide a second indicator that the DUT is definitely executing the second loop, the toggling of `PIN_STATUS_2` can be tested.

We investigate the effect of a negative voltage glitch, as this method has been reported to be successful for other microcontrollers [25, 39]. In our framework, the following parameters (cf. Fig. 23) can be varied:

– The glitch offset t_{offset} with respect to the trigger rising edge,
– the glitch width t_{width}, and
– the glitch voltage level V_{low}, i.e., the value to which the supply voltage is temporarily reduced to.

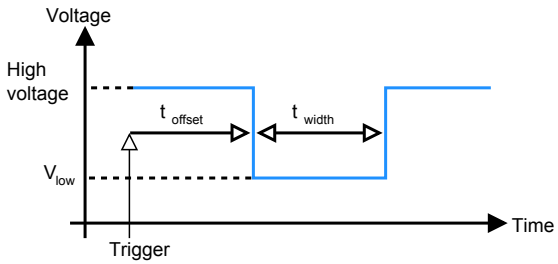

Fig. 23. Parameters characterising a single negative power glitch.

In order to systematically determine the settings that lead to the desired effect, we implemented an application with a "sweep mode" that consecutively tests all combinations in a certain range for each of the values. This way, the device is fully profiled for all possible parameters without human interaction. To demonstrate the integration of fault injection with the measurement framework and to be able to analyse the fault effect afterwards, the application also records oscilloscope traces of the voltage glitch on the supply rail and the state of the status pins.

Results Using the data gathered by the parameter sweep, three effects can be identified:

1. Injection not successful, i.e., `PIN_STATUS` remains set and the first loop is not left, see Fig. 25.
2. The device is reset, resulting in `PIN_STATUS` to be set low for a short time (during the initialization instructions) and then high again when the first loop is entered, see Fig. 24.
3. The desired fault is injected, i.e., `PIN_STATUS` stays low permanently, indicating that the microcontroller executes the second loop, see Fig. 26.

Figures 25, 24 and 26 display example oscilloscope traces for each outcome. For case 3, Fig. 27 additionally shows the toggling waveform on `PIN_STATUS_2` in `loop2` after a successful instruction skip fault.

Based on these experiments, we conclude that power glitch attacks to skip instructions on the PIC16F687 are possible, provided that the voltage is reduced to a value within a region from 1.65 V to 1.73 V and the fault occurs at the correct point in time, where a rather large set of timing parameters turned out to work for the scenario of leaving an endless loop.

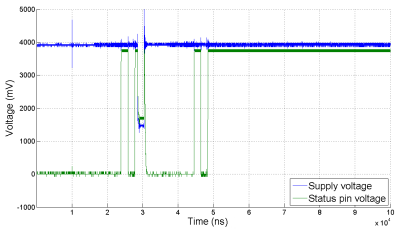

Fig. 24. Waveform of reset after fault injection on PIC16F687.

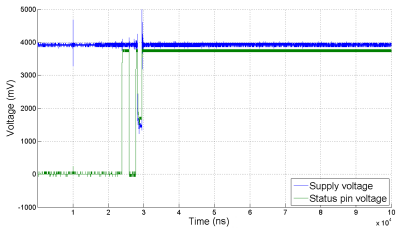

Fig. 25. Waveform of unsuccessful fault injection on PIC16F687.

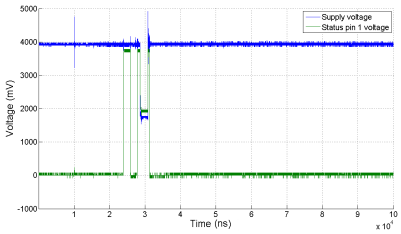

Fig. 26. Waveform of successful fault injection on PIC16F687, displaying `PIN_STATUS`.

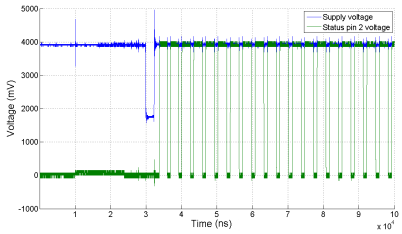

Fig. 27. Waveform of successful fault injection on PIC16F687, displaying `PIN_STATUS_2`.

Multiple Faults On the basis of the results of the previous section, the scenario is now extended to multiple fault injection. The test code has been modified: The first loop remains unchanged, while in the second loop the toggling of `PIN_STATUS_2` has been removed. The third loop catches the successful exit from `loop1` and `loop2`, indicating this condition by setting `PIN_STATUS_3` and toggling `PIN_STATUS_2` as additional criterion for `loop3`.

Thus, if the first two loops can be skipped using two successive faults, this condition is detected by checking for `PIN_STATUS = 0` and `PIN_STATUS_3 = 1`, which can again be accomplished automatically using the FPGA user I/O pins. For illustration, we recorded the waveform on `PIN_STATUS_2`, as the toggling provides visual evidence that the microcontroller is indeed executing `loop3`. The two successive negative voltage glitches are now characterised by 6 parameters, summarised in Fig. 28:

- The first glitch offset $t_{offset,1}$ with respect to the trigger rising edge,
- the first glitch width $t_{width,1}$,
- the first glitch voltage level $V_{low,1}$,
- the second glitch offset $t_{offset,2}$ with respect to the end of the first glitch,
- the second glitch width $t_{width,2}$, and
- the second glitch voltage level $V_{low,2}$.

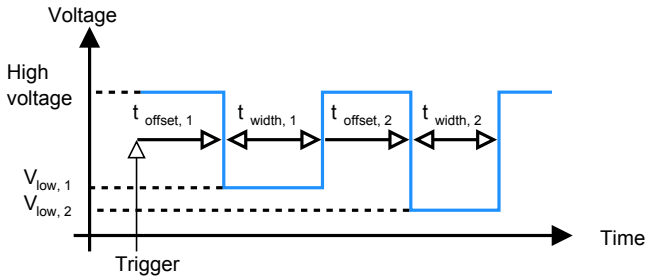

Fig. 28. Parameters characterising a negative double power glitch.

To reduce the overhead for the search through all parameter combinations, the first glitch is fixed based on a setting that led to a successful fault in the single fault scenario. Moreover, the low voltage level is set equal for both glitches.

Results By conducting a parameter sweep with the first pulse fixed, we were able to skip both `loop1` and afterwards `loop2`, resulting in the waveform in Fig. 29. Having profiled the device, we could repeat the experiments with identical parameters and reliably perform the fault injection, thereby achieving a success rate close to 100 %.

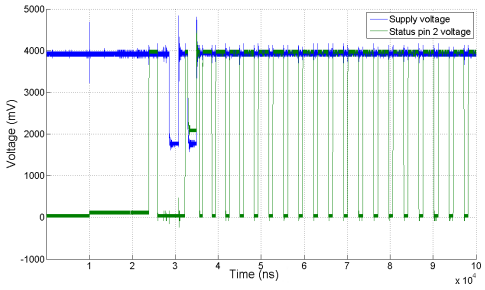

Fig. 29. Waveform of successful multiple fault injections on a PIC16F687, displaying `PIN_STATUS_2`.

Implications Depending on the cryptographic primitive and its actual implementation (and on the adversary's knowledge about the implementation), the injection of power faults as practically demonstrated above can have various dramatic effects on the security. Some examples are listed below, while the list could be vastly extended.

– In the case of a block cipher, e.g., AES, that is normally executed for several rounds to produce a cryptographically secure output, it is conceivable

to skip the appropriate jump instruction that will execute the next round of the cipher: Having the plaintext and the output after one round, it is straightforward to mathematically conclude to the secret key used for the encryption.
- If a device is protected against power analysis with masking, the injection of a fault [3], e.g, while the mask is initialised, may allow to circumvent the countermeasure. If the mask remains set to zeroes it will have no protecting effect and a subsequent power analysis will become possible by this combination of active and passive side-channel analysis, which is easily possible with our proposed setup.
- Since inducing multiple faults is relatively simple, as demonstrated above, even common countermeasures against fault injection could be circumvented: Often two computations of, for example, the same exponentiation are carried out and the results are compared. If they are equal, the device assumes that no fault has occurred. To spoof the protection mechanism, hence one fault needs to be induced during the computation, and a second fault during the comparison.
- An induced fault could also allow for reverse-engineering of a (secret) program code that is executed by the microcontroller: Many devices feature a read-out protection, i.e., if a protection bit is set to one in the internal circuit of the microcontroller, the program code cannot be read out. A fault injected at the right moment in time, whilst trying to read out the device, could make this bit read zero, disabling the code protection mechanism.

Since all tests can be carried out automatically with varying parameters, and the success of the fault injection is automatically detected (the feedback via the data acquisition module even allows to determine, whether a reset of the DUT needs to be triggered), no human interaction is required. Thus, it is conceivable to perform a thorough profiling and find the correct points in time for inducing faults even in a black-box scenario.

7 Conclusion

In this article, we present a versatile framework, allowing for implementation attacks on virtually all types of cryptographic devices. Our setup provides functions for the automatic profiling and security evaluation of a cryptographic (or non-cryptographic) device, both with respect to side-channel analysis, i.e., the evaluation of EM or power measurements, and for fault injection, including the determination of parameters required to actively attack a black-box device.

The introduced functional units cover amongst others the analysis of (contactless) smartcards, RFIDs, microcontrollers, ASICs, FPGAs and mobile computing devices. After discussing models and possible effects of inducing faults, we describe our customised self-built hardware modules for generating faults based on power glitches, clock variations, as well as optical and EM fault injections. To our knowledge, we propose the first circuit for automatically inducing faults

based on sparks in cryptographic devices published in the literature. The modules allow for automatically profiling the parameters and the strength of faults required for a particular device, as a function of the data processed and by using the data acquisition module as a feedback channel. The data acquisition module itself is based on a USB oscilloscope and enables inter alia for side-channel analysis based on the EM field, current consumption, and timing of the DUT.

We exemplarily demonstrate the effectiveness of the system by profiling a contactless smartcard and identify the appropriate leakage model. On this basis, we perform the first reported successful full key-recovery of a commercial cryptographic RFID employing Triple-DES by means of DEMA. As a second practical result, we prove the feasibility of multiple successive fault injections on a widespread PIC microcontroller using power glitches. This has severe implications with respect to the effectiveness of many countermeasures that often — due to overrating the efforts required for such an attack — protect only against one single fault during a computation. As all parts of our framework may operate independently of each other, powerful combined active and passive attacks are enabled. These attacks, e.g., circumventing countermeasures against power analysis by injecting faults, have been theoretically proposed in the literature, but so far there has been a lack of practical results regarding their feasibility.

We conclude that most implementation attacks, including the injection of multiple faults, can be conducted with a low-cost, public domain lab setup as described in this article. The common belief that highly sophisticated and expensive equipment is required is proven to be wrong: State-of-the-art implementation cryptanalysis can be performed by anyone who has a sufficient know-how about the attacks.

References

1. ISO 7816 Identification Cards - Integrated Circuit Cards with Contacts, 2004.
2. D. Agrawal, B. Archambeault, J. Rao, and P. Rohatgi. The EM Side-Channel(s). In B. Kaliski, e. Koç, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2002*, volume 2523 of *Lecture Notes in Computer Science*, pages 29–45. Springer Berlin / Heidelberg, 2003.
3. F. Amiel, K. Villegas, B. Feix, and L. Marcel. Passive and active combined attacks: Combining fault attacks and side channel analysis. In *FDTC '07: Proceedings of the Workshop on Fault Diagnosis and Tolerance in Cryptography*, pages 92–102, Washington, DC, USA, 2007. IEEE Computer Society.
4. Analog Devices, Inc. *AD8058 Dual, High Performance Voltage Feedback, 325 MHz Amplifier Datasheet*, 2003.
5. Analog Devices, Inc. *AD9708 8-Bit, 100 MSPS+ TxDAC D/A Converter Datasheet*, 2009.
6. Atmel. ATMega32 Data Sheet. `www.atmel.com`.
7. Atmel. Datasheet of Read/Write Base Station U2270B. `www.atmel.com`, 2008.
8. E. Brier, C. Clavier, and F. Olivier. Correlation Power Analysis with a Leakage Model. In M. Joye and J.-J. Quisquater, editors, *Cryptographic Hardware and Embedded Systems - CHES 2004*, volume 3156 of *Lecture Notes in Computer Science*, pages 16–29. Springer, 2004.

9. D. Carluccio, K. Lemke, and C. Paar. Electromagnetic Side Channel Analysis of a Contactless Smart Card: First Results. RFIDSec05 Workshop on RFID and Lightweight Crypto, July 2005. http://events.iaik.tugraz.at/RFIDandLightweightCrypto05/RFID-SlidesandProceedings/Carluccio-EMSideChannel.pdf.
10. C. C. Club. RFID Zapper, 2005.
11. D. Corson. Comparing 8-bit Microcontrollers for Ultra-low-power Applications. pages 3, table 1, October 2005.
12. J. J. A. Fournier, S. Moore, H. Li, R. Mullins, and G. Taylor. Security Evaluation of Asynchronous Circuits. pages 137–151, 2003.
13. C. Giraud and H. Thiebeauld. A Survey on Fault Attacks. In J.-J. Quisquater, P. Paradinas, Y. Deswarte, and A. A. E. Kalam, editors, *CARDIS*, pages 159–176. Kluwer, 2004.
14. H. B.-E. Hamid, H. Choukri, D. N. M. Tunstall, and C. Whelan. The Sorcerer's Apprentice Guide to Fault Attacks. 2004.
15. H. Handschuh. Contactless Technology Security Issues. Information security bulletin, volume 9, 2004. http://www.chi-publishing.com/samples/ISB0903HH.pdf.
16. H. Handschuh, P. Paillier, and J. Stern. Probing Attacks on Tamper-Resistant Devices. In *CHES*, pages 303–315, 1999.
17. M. Hutter, J.-M. Schmidt, and T. Plos. RFID and Its Vulnerability to Faults. In E. Oswald and P. Rohatgi, editors, *Cryptographic Hardware and Embedded Systems Ű CHES 2008*, volume 5154 of *Lecture Notes in Computer Science*, pages 363–379. Springer Berlin / Heidelberg, 2008.
18. International Organization for Standardization. *ISO/IEC 14443-3: Identification Cards - Contactless Integrated Circuit(s) Cards - Proximity Cards - Part 3: Initialization and Anticollision*, 1st edition, February 2001.
19. International Organization for Standardization. *ISO/IEC 14443-4: Identification cards - Contactless Integrated Circuit(s) Cards - Proximity Cards - Part 4: Transmission Protocol*, 1st edition, February 2001.
20. ISO/IEC 14443. Identification cards - Contactless integrated circuit(s) cards - Proximity cards - Part 1-4. www.iso.ch, 2001.
21. T. Kasper. Embedded Security Analysis of RFID Devices. Master's thesis, Ruhr Universität Bochum, 2006.
22. T. Kasper, D. Carluccio, and C. Paar. An Embedded System for Practical Security Analysis of Contactless Smartcards. In *WISTP*, volume 4462 of *LNCS*, pages 150–160. Springer, 2007.
23. T. Kasper, D. Oswald, and C. Paar. EM Side-Channel Attacks on Commercial Contactless Smartcards using Low-Cost Equipment. In *Proceedings of WISA 2009*, to appear in Springer LNCS, September 2009.
24. T. Kasper, M. Silbermann, and C. Paar. All You Can Eat or Breaking a Real-World Contactless Payment System. In *Financial Cryptography and Data Security 2010*, volume 6052 of *LNCS*, pages 343–350. Springer, 2010.
25. C. H. Kim and J.-J. Quisquater. Fault Attacks for CRT Based RSA: New Attacks, New Results, and New Countermeasures. In *WISTP*, pages 215–228, 2007.
26. O. Kömmerling and M. G. Kuhn. Design Principles for Tamper-Resistant Smartcard Processors. pages 9–20, 1999.
27. P. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In N. Koblitz, editor, *Proceedings of CRYPTO 1996*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113. Springer-Verlag, 1996.

28. P. Kocher, J. Jaffe, and B. Jun. Differential Power Analysis. In M. J. Wiener, editor, *Proceedings of CRYPTO 1999*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer-Verlag, 1999.
29. T. Kugelstadt. *Op Amps for Everyone*, chapter 14 - Interfacing D/A Converters to Loads, page 239. Texas Instruments, 2nd edition, 2003.
30. S. Mangard, E. Oswald, and T. Popp. *Power Analysis Attacks: Revealing the Secrets of Smart Cards*. Springer-Verlag, Secaucus, NJ, USA, 2007.
31. M. Melito. Application Note 484/1293, Car Ignition with IGBTs, 1999.
32. Microchip Technology Inc. *PIC16F631/677/685/687/689/690 Data Sheet 20-Pin Flash-Based, 8-Bit CMOS Microcontrollers with nanoWatt Technology*, 2008.
33. K. Nohl, D. Evans, Starbug, and H. Plötz. Reverse-Engineering a Cryptographic RFID Tag. In P. C. van Oorschot, editor, *USENIX Security Symposium*, pages 185–194, 2008.
34. NXP. *Data Sheet of Mifare Classic 4k chip MF1ICS70*, 2008.
35. Y. Oren and A. Shamir. Power Analysis of RFID Tags. `http://www.wisdom.weizmann.ac.il/~yossio/rfid/`.
36. T. Plos. Susceptibility of UHF RFID Tags to Electromagnetic Analysis. In T. Malkin, editor, *Topics in Cryptology - CT-RSA 2008*, volume 4964 of *Lecture Notes in Computer Science*, pages 288–300. Springer Berlin / Heidelberg, 2008.
37. Potato Semiconductor Corporation. *PO74G08A Quadruple 2-input positive AND gate*, 2009.
38. J.-M. Schmidt. Differential Fault Analysis - Final Report. Technical report, TU Graz, June 2008.
39. J.-M. Schmidt and C. Herbst. A Practical Fault Attack on Square and Multiply. In *Proc. 5th Workshop on Fault Diagnosis and Tolerance in Cryptography FDTC '08*, pages 53–58, Aug. 10–10, 2008.
40. K. S. Shanmugam. *Digital & Analog Communication Systems*, chapter 8.3.2. Wiley-India, 2006.
41. ST Microelectronics. Data Sheet for STP10NK50Z, N-Channel Zener-Protected MOSFET, 2005.
42. J. Waddle and D. Wagner. Fault Attacks on Dual-Rail Encoded Systems. *Computer Security Applications Conference, Annual*, pages 483–494, 2005.
43. Xilinx Inc. *PicoBlaze 8-bit Embedded Microcontroller User Guide*, v. 1.1.2 edition, June 2008.
44. Xilinx Inc. *Spartan-3 FPGA Starter Kit Board User Guide*, v 1.2 edition, June 2008.
45. Xilinx Inc. PicoBlaze User Resources . web resource, 2009.