Practical optical fault injection on secure microcontrollers

Jasper G. J. van Woudenberg, Marc F. Witteman, Federico Menarini

Riscure BV, 2628XJ Delft, The Netherlands
{vanwoudenberg,witteman,menarini}@riscure.com

Abstract. In this paper we detail the latest developments regarding optical fault injection on secure microcontrollers. On these targets, a combination of countermeasures makes fault injection less than trivial. We develop fault injection methods to show experimentally that protected smart cards are still vulnerable. We perform power signal guided fault injection, using a triggering mechanism based on real-time pattern recognition. Furthermore, the use of jitter-free diode lasers shows current countermeasures may be inadequate for the near future.
Keywords — optical fault injection, differential fault analysis, pattern based trigger, countermeasure, diode laser, secure microcontroller.

1 Introduction

Secure microcontrollers, such as smart cards, store a wide variety of sensitive assets: this ranges from PIN codes in banking cards, to subscriber identities in SIM cards, and to fingerprints in biometric e-passports. These assets are highly valuable, and therefore the main function of the secure microcontroller is to protect their integrity and confidentiality.

There are a number of ways attackers try to subvert the protection mechanisms built into cards. Attacks can be categorized as active or passive. Active attacks operate the card intentionally outside of its normal operating conditions. Passive attacks operate within normal specifications, but try to find leakage of secret information. Further, attacks can be non-invasive, semi-invasive or fully invasive. Non-invasive attacks do not physically alter the card, semi-invasive attacks may involve chip decapsulation, and fully invasive attacks may entail delayering or probing the chip's surface. Differential power analysis [Koch99] is an example of a passive non-invasive attack. Fault injection, the topic of this paper, is an example of an active attack that can be performed at all levels of invasiveness.

Fault injection (or perturbation) attacks involve actively manipulating a chip in order to cause a transient fault during the execution of some process. The goal is to circumvent the protection of its assets. A fault can for instance allow bypassing security condition checks, such as PIN correctness. Faults can also be used for Differential Fault Analysis. An extreme example of DFA is the RSA CRT Bellcore attack [Bone01], which allows retrieval of the full RSA private exponent from one faulty computation. Other attacks allow retrieving keys of a range of public and secret key algorithms, including DES and AES [Biha97,Karp04].

Manufacturers of secure devices are aware of fault injection threats and typically implement a range of countermeasures to mitigate the risks. At the same time adversaries improve attack methods to challenge these countermeasures, leading to an interaction that results in a continuous improvement of card security.

In this article we mainly focus on *optical fault injection*, a semi-invasive fault injection attack involving a high-intensity laser inducting faults. Since Skorobogatov's et al.'s public introduction of this fault injection method [Skor03], secure microcontrollers contain increasingly strong countermeasures. Attacks such as [Skor10] aim at specific targets, in this case memory write and erase operations. The aim of this paper is to more generally describe the tools and methods required to analyze state-of-the-art secure microcontrollers using optical fault injection. Note that most of the concepts applied to optical fault injection extend to other types of glitching.

The remainder of this paper is organized as follows. In the next section we give some background on fault injection. After this we continue to describe common countermeasures found in modern secure microcontrollers. Then we describe the fault injection hardware needed to theoretically overcome these countermeasures, and give several experimental results. We conclude with some final remarks.

2 Background

There are several methods for inducing a fault in a computation. On the non-invasive side, we can perform clock or voltage glitching [Chou06]. Smart cards have external power and clock supplies. Both of these contacts can be operated outside of their specified ranges, possibly causing faults in the pro-

cess running on the card. The cost of the tools is low, although this also holds for the success rate on modern secure microcontrollers.

On the fully invasive side, there is active fault injection by microprobing. For this a card needs to be prepared: the chip needs to be decapped, the passivation layer removed, and the shielding needs to be circumvented. Although not impossible, this is a very laborious process [Tarn10]. Due to the cost of this method and the tools required, we do not consider this to be a very practical attack path.

Optical fault injection, however, requires only minimal chip preparation: access needs to be obtained to either the front or the back side of the chip. This can be accomplished by decapping or smart card contact pad removal. On recent secure microcontrollers the cost is increasing due to the higher demands on accuracy; however, in terms of time and monetary cost it is still quite feasible for a reasonably equipped adversary.

The remainder of this section will detail several ways of injecting faults and how to use any successful attempts.

2.1 Voltage and clock glitching

With clock glitching, the device clock is temporarily accelerated by introducing one or several short pulses in the external clock provided to the smart card. This has the effect of possibly introducing a fault in the execution of an instruction; when e.g. a CPU is reading a memory cell at the time of a glitch, the results may be read before the data is stable on the memory bus. This results in reading a wrong value. Similarly, a voltage glitch may result in wrong values being read from memory at the time of the glitch.

Another effect that can occur is that an instruction is fetched from memory, but its execution is never completed: the fast clock has caused the next instruction to already be fetched.

We find most modern secure microcontrollers have adequate protection against these types of attacks. The ranges these controllers are expected to operate within are well defined [iso06], and can be verified by the chip. Secure microcontrollers also run most of their operations based on an internally generated clock, and therefore they are not as sensitive to outside clock glitches as microcontrollers running on an externally generated clock.

The countermeasures focus on both the prevention of glitches by filtering the input, and on detection by continuously monitoring these interfaces. If out-of-specification input is detect, the card may respond to this event as described in section 3.

2.2 Optical fault injection

Optical glitches are generated with a strong light source, e.g. a photo flash or a laser beam. As semiconductors are inherently sensitive to light it is possible to switch transistors when exposed to an optical pulse. When using a precision stage and a focused laser beam, it is possible to accurately target certain regions of a chip. Although this creates an extra two dimensions of parameter value search space (X and Y coordinates), it allows on very specific focusing on e.g. memory decoders, CPU or cryptographic components. Also photosensitive countermeasures can be avoided. Furthermore, spot size and photon wave length are other relevant parameters.

In terms of preparation the chip die needs to be optically exposed. The front side is the side of the metal layers and transistors, which are covered in an epoxy. This side typically requires removing this epoxy (decapping), although for some chips even this is not necessary as they have transparent epoxy covering them. The back side of the chip is the side of the substrate, which typically requires removing the center part of the smart card contact pads in order to be exposed. Sometimes depackaging and rebonding is necessary before it can be accessed.

Because of its minimal preparation required, optical fault injection is considered semi-invasive. Despite of this drawback the method is the most successful smart card perturbation attack as countermeasures against it are not easily implemented [Skor03].

2.3 Exploitation

Fault injection is employed to achieve various effects. In this section we highlight a few of them: access bypass, memory dumping, (partial) key nulling, differential fault analysis, and influencing (side channel analysis) countermeasures.

Access bypass is achieved when the attacker glitches a sensitive decision (e.g. an authentication result), resulting in more privileges. With these

privileges the attacker may access sensitive data. An example would be glitching a PIN verification, after which a banking card will sign transaction requests.

To obtain a memory dump the attacker glitches the device while it is transmitting data. Due to the glitch, data is transmitted from the wrong memory location, or excessive data is dumped. In both cases confidential data may be leaked. Older smart cards were sometimes vulnerable to this attack during transmission of their ATR, in extreme cases leading to full memory dumps.

Partial key nulling is an attack whereby an attacker sets a fraction of a secret key to all 0 bits (or all 1), and does not affect a few bytes of the key. With knowledge of the rest of the key, those remaining few bytes can be bruteforced if the input and output data is known. Full key nulling can be interesting in attacks on secret key protocols: if a secret key is used for e.g. authentication, forcing a key to a known value allows an authentication protocol to succeed without knowledge of the actual secret key.

Differential Fault Analysis (DFA) can be performed after glitching a cryptographic operation, resulting in corrupted signatures or cryptograms. With mathematical analysis these corrupted data can be used to extract a secret or private key. An example of a single fault DFA is the RSA/CRT Bellcore attack [Bone01], which allows retrieval of the full RSA private exponent from one faulty computation. Other attacks allow retrieving keys of a range of public and secret key algorithms, including DES and AES [Biha97,Karp04].

Finally, it may be possible to enhance side channel analysis or fault injection by disabling or influencing countermeasures. As some countermeasures are configurable, they are switched on at some point during bootup. By glitching it may be possible to skip switching these on, or e.g. disturbe the balance of random number generators.

3 Countermeasures

As the attacks evolve, so do their countermeasures. Chip manufacturers and card vendors are aware of the threats posed by fault injection. Both therefore implement a combination of countermeasures, as the most secure cards includes a mix of hardware and software features [Komm99,Witt08].

Of course, every countermeasure comes at a price. The main trade-offs lie in manufacturing cost versus hardware countermeasures, card performance versus software countermeasures, and countermeasure sensitiveness versus card failure rate. It is common knowledge perfect security does not exist. However, even the best practically attainable security is not reached in order to have a card that is cost-effective, performs well, and does not fail constantly. The countermeasures described in this section therefore have the goal of making attacks sufficiently expensive, not impossible.

3.1 Hardware barriers

Chip manufacturers use physical barriers, like shields to protect the chip from malicious manipulation. Shields can be passive or active. A passive shield is a metal layer covering large parts of the chip surface to prevent physical access with probes or optical beams. Passive shields may not completely cover the chip. They often consist of a matrix of smaller shields placed at a small distance, as this makes the chip less sensitive to breaking due to temperature fluctuations. Passive shields can typically be removed without the chip detecting this.

An active shield consists of a wire mesh that runs life signals over the chip surface and detects any interruption of the wire. Active shields offer a better protection against modification, but are more transparent to light. These shields are intended to detect physical tampering, but as shown in [Tarn10], a well-funded and highly determined attacker can reroute wires around a hole that he has made in the shield.

3.2 Hardware sensors

Modern smart card chips include a wide array of sensors to detect anomalies, including voltage, clock, photon and temperature sensors. With these sensors configured correctly it should no longer be possible to use e.g. voltage spikes to introduce faults. However, strict sensor settings may impact operation reliability: a cell phone with almost flat battery for instance may not be able to keep the SIM card energy resource within the standardized range. Therefore manufacturers or card vendor may choose rather permissive sensor settings that leave a vulnerability to glitching.

Optical sensors attempt to detect physical opening of the device, and catch scanning laser beams [Dero07]. Although these sensors can be effective against laser scanning, it is impossible to protect each transistor. An attacker who has found the right position to hit with a laser succeeds if he manages to make the optical spot size on the chip small enough to miss the nearest sensor.

3.3 Other hardware features

Smart card chip features like internal and drifting clocks are not primarily designed as fault injection countermeasures, but do make perturbation more difficult: the clock cannot easily be manipulated and the exact timing of the instruction to hit is difficult to predict due to the clock instability. Also internal clocks tend to run much faster (>30 MHz) than external clocks (~4MHz). Consequently, successful glitches should become much shorter to avoid hitting multiple instructions and getting unpredictable or useless results.

3.4 Software countermeasures

Perturbation aware software aims at both decreasing the probability of a fault injection and detecting the fault by verification of computed intermediate results.

Decreasing the fault probability is typically achieved through random delays and process ordering. Without some form of synchronization, an attacker may need many repetitions of his experiments before he hits the right instruction. However, the insertion of random delays leads to a significant execution slowdown.

Verification involves checking the validity of a decision, address or data, and attempts to prevent abuse of a successful fault. The software may double check all security sensitive decisions and create verification checksums for all sensitive data. Also the program flow must be protected to avoid 'code hijacking' when the program counter is glitched during branch or return instructions. Finally, cryptographic signatures will always be verified before transmission to avoid differential fault analysis. Verification is expensive since checking the effect of one instruction costs multiple instructions.

Whenever verification fails, or a hardware countermeasure is triggered, a software response can be evoked. A device can move to a locked state, overwrite its secret data or even terminate. This obviously needs to be tuned not to have devices fail as a result of alleged perturbations that could be caused by non-hostile phenomena.

4 Optical fault injection on modern secure microcontrollers

Given the range of countermeasures implemented, the bar for modern secure microcontrollers is relatively high compared to non-secure microcontrollers or older technology. It is necessary to be able to very precisely focus on a certain area of a chip in order to focus on a particular sensitive area, avoiding any sensors or triggering unwanted faults. Furthermore, the laser should only be switched on during the execution of a specific instruction or instructions, sometimes only during a part of a clock cycle. Due to verification countermeasures, it is also of importance to be able to inject multiple faults during a computation.

Under these conditions, optical fault injection on these targets will be increasingly unsuccessful with tools from the earlier days, like simple flash lights and laser pens [Skor03], or even laser cutters. The two former have neither position nor time accuracy, and the latter lacks in time accuracy.

In this section we discuss the fault injection setup we designed to overcome these shortcomings and adhere to the specifications needed for optical fault injection on modern secure microcontrollers.

4.1 Triggering

As we intend to trigger the laser on one or multiple instructions, we need to have a trigger that synchronizes to these instructions. Traditionally, time based triggers have been used. They use a hardware-based timer that triggers a fault at a programmable delay after a communication event. Due to drifting clocks and variable delay countermeasures the probability of hitting a selected target instruction decreases. This can be acceptable in situations where injection attempts can be repeated indefinitely without penalty or situations where many target instructions would be suitable fault candidates. The latter can be true for Differential Fault Analysis (especially for DFA on RSA in CRT mode). However, if the device implements consistent software verification it

is no longer practical to inject faults using a time based trigger, as the jitter between the communication event and the target instruction is relatively large.

Pattern based triggering is an alternative to time-based triggering. With this technique a signal is observed and a real-time trigger is produced when the analog signal corresponds to a preprogrammed reference signal. We implemented pattern based triggering using an FPGA board with fast A/D sampling. The FPGA code performs continuous comparison of an incoming signal with the reference signal, and produces a trigger when the Sum of Absolute Differences drops below a programmable threshold.

Although the FPGA board can sample at 100 MHz, and performs array comparison on a 10 ns time base, this may not be sufficient for noisy and signals with high frequency patterns. Therefore we included a frequency conversion filter using a narrow band filter. The narrow band filter is based on a common architecture used for demodulation in AM radio receivers, where a carrier wave is removed from a modulated signal and an envelope signal remains. This envelope signal is of a much lower frequency, but represents patterns occurring at a higher frequency. With this filter it is possible to sample, compare and trigger on signal features up to 400 MHz.

For fault injection, we use this device to synchronize to the power signal coming from a chip. As this signal typically shows enough variation to identify different process steps, it is possible to find a unique pattern just before the fault injection target instruction. After loading this reference pattern, we can perform a time-based trigger relative to the pattern based trigger in order to achieve a precise fault injection. The effective jitter relative to the target instruction now only depends on the timing variation introduced by the card in the short period between the instruction and the reference area, and on the jitter in the pattern based trigger (max 10ns).

4.2 Optical setup

Besides an accurate trigger, we also need a low-jitter laser to respond to this trigger. Furthermore, it needs to have a spot size that allows triggering only small areas of the chip. We chose to use diode lasers as they provide very accurate timing with sub-nanosecond scale jitter. Diode lasers can switch very quickly (up to 25 MHz repetition rate), which offers the possibility of multiglitching. This means that consecutive instructions can be manipulated, and verification countermeasures may be defeated. The diode laser is mounted in a microscope setup, with a camera for visual positioning, an XY stage for moving the target, and 5×, 20×, and 50× objectives for generating different spot sizes.

Although the power of diode lasers used to be too low for shield or substrate penetration, developments have resulted in powerful devices. The newest generation of diode lasers (multi-mode) is sufficiently powerful ($>$ 10 Watts). Since these diode lasers produce a non-circular and divergent light source, extra lenses are needed to correct the shape and divergence and shrink the size of the light spot on the die. A further reduction is achieved with objectives that reduce the spot size to a minimum of a few μm. This is limited by the physics of light: a spot can never have smaller dimensions than the wavelength of its photons (for common lasers in the range 500-1000nm). With feature sizes moving below 100nm, even with the smallest possible spot size we will be hitting multiple transistors. From practice, we observe this size is small enough to stay away from sensors when targeting specific groups of transistors on a chip.

When attacking a chip, the attacker can choose to target the front side or back side of the chip (see Figure 1). The transistors are positioned at the front side, but they may be difficult to reach due to metal shields (although gaps in the shield may allow some light to pass). The back side of the chip is formed by the substrate, a slice of silicon that acts as a frame for the chip. This substrate needs to be penetrated before the transistors are reached.

A back side attack is preferably performed with light wavelength for which silicon is semi-transparent, i.e. longer than about 1000 nm. Front side attacks can be performed with many wavelengths, but shorter wavelength light is more suitable than infrared as transistors switch easier due to the higher energy content. Together with the limited choice between different wavelength for diodes, we use 808nm for front side attacks and 1064nm for back side attacks.

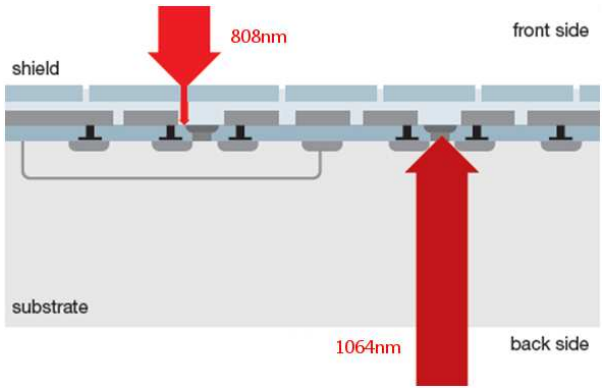

Fig. 1. Front and back side attacks

4.3 Parameter determination

In order to perform a successful laser fault injection attack, several parameters need to be found. Such parameters include the location on the chip, the time delay between a synchronization point and the target instruction, the duration of the laser pulse, the intensity of the laser, the spot size, the number of pulses, and their mutual distance. Full exploration of the whole space is an exponential problem and therefore is for even small parameter ranges practically infeasible.

A practical approach lies in taking a divide-and-conquer approach by tuning parameters individually, rather than as a complete set. Several experiments and other sources of information can help individual tuning:

- The timing and synchronization points for each target instruction can be derived by taking a power trace of the execution. Especially having programmable cards where one can compare a power trace with the target operation, and one without, can help in pinpointing the exact timing.
- By attempting injections and observing the effect on the process by analyzing the power use and card output, it is often possible to establish the effect of a single fault. Once the timing for the first fault is found, the search for the proper offsets of subsequent faults can continue.
- The location of the laser pulse on the chip can be derived by visual narrowing down: if the target is a hardware DES, there is likely no point in targeting the center of the ROM area. This should be used with caution; we have experienced finding unexpected target areas that have yielded successful injections (e.g. attacking the DES execution by targeting the key loading from EEPROM). After roughly locating the area, it is possible to attempt an injection at each location within this area.
- The intensity of the laser is determined by attempting to inject fault at increasing laser power. Note that different areas may have different sensitivities.
- The spot size is determined by the choice of objective, and with the 50× objective is $6 \times 1.4\mu$m. The *effective spot size* is actually larger than the projected laser beam due to scattering of the photons between metal layers and through the substrate. We typically start with the 5× objective, and only go to smaller sizes if necessary as a smaller spot implies more precise scanning to find a sensitive spot.
  There is some remaining risk here that sensors may be activated. This can be mitigated by smaller spot sizes, or by reacting to 'tripping' a sensor, as is explained in section 5.2.
- The duration of one laser pulse is typically related to the clock frequency of the attacked chip. We may start with longer pulses to observe any deviating response, and narrow down the duration to smaller pulses to achieve specific responses.

Obviously, the injection of a fault is only possible when all parameters have been found correctly. As every target is different, the points above do not guarantee success. They are typically applied iteratively, narrowing down on the right ranges by looking at the feedback from the power measurements and card responses. Currently, it is an open research question on how to optimize this process.

5 Experiments

Using the system described in the previous sections, we perform several experiments to verify whether the described countermeasures can be dealt with.

5.1 Card 1

This card is a programmable card without OS. Due to its lack of any hardware countermeasures, it is a good card to perform baseline experiments

against. We use this card to verify a few capabilities: timing stability of the laser and trigger generator, multiglitching capabilities and the possibility of performing a back side attack.

We program it with an application that performs a PIN verification twice: only if both verifications succeed, access is granted. For testing purposes we have the application return which verification has succeeded in its response code, allowing us to discern the effects of each of our laser pulses.

We remove the center part of the smart card contact pads to expose the substrate of the chip. Then, the smart card is inserted into a reader with programmable hardware timers and fast pulse generators, which will be used to pulse the laser at fixed intervals after the card starts processing. For the back side of the chip we use a 1064nm laser with a pulse rate of 25MHz. The laser is capable of 20W, but we operate it at 50% power. By using the 5× objective we get a spot size of approximately $60 \times 14\mu$m, but due to the high intensity laser the effective spot size will be larger due to photon scattering on the chip. The card is running at 1MHz, and from previous experiments we know the card can handle pulses of the duration of its clock cycle with this intensity.

As this card has a single CPU, we know the PIN verification is happening at approximately the same location as e.g. the code sending its ATR. The ATR is the first response a card gives on the I/O line after power up. We perform a fault injection on each location in a 20×20 grid over the full chip and find several locations where the card is not responding with a full ATR, or with a corrupted ATR. This gives us candidate locations at which the CPU is sensitive, and where we may find the right timing for injecting faults in the PIN verification.

We first take a power trace and compare it with the source code to determine the a range of about 100 clock cycles (100μs) starting at time $t = 600\mu$s in which the first PIN verification should happen. For each injection attempt we present the card with a faulty PIN code. For several candidate locations we pulse the laser at $t + k, k \in \{0, 1, \ldots, 99\}$, and observe the response code of the application. For a particular location, we see at $k = 72\mu$s that the application responds the first PIN verification has succeeded, but the second one has failed. Note this can also be observed in the power trace by a slightly longer processing time.

Fixing this location and first target timing, we proceed in finding the timing of the second target. We pulse the laser for a second time at $t + k + i$ for $i = 1, 2, \ldots$ until we find that at $i = 33$ the card responds a successful PIN verification.

Next, we fix all parameters to determine the jitter of the system as a whole. As the card has a stable clock and no countermeasures, the fault injection probability depends only on the setup. For 1000 consecutive injections we find the card always returns the PIN has successfully been verified. When k or i is changed by 1μs, the card no longer reports a successful verification. Also, exchanging the 1064nm laser for a 14W 808nm laser shows no more successful faults.

This experiments shows that in order to attack even this unprotected card, a time resolution of 1μs is necessary for each pulse, which is met by the diode lasers and the trigger device. Furthermore it shows that 1064nm is a necessity for back side attacks, as even at higher power 808nm does not sufficiently penetrate the substrate.

5.2 Card 2

This card is programmable, is OS-based and runs off an unstable internal clock, and has detection of optical faults. It has transparent epoxy and no shields covering the front side. We use this card for testing front side attacks, and for testing pattern based triggering. Also, we will be targeting a hardware DES accelerator running with an unstable clock of around 30MHz.

Card preparation is trivial: a small hole in the plastic covering the chip is required. The laser is able to penetrate the transparent epoxy. We load an applet performing a DES encryption on the hardware accelerator with a configurable key, and with the supplied input data. The applet returns the encrypted output.

The DES execution can be observed in the power trace. However, the timing is inconsistent due to the unstable internal clock. We therefore find a synchronization pattern before the DES execution, and use this as a pattern to trigger the power measurement. Now, the DES execution can be seen in Figure 2 between 90μs and 120μs. The trigger pattern is repeated before and after the DES at a relatively consistent interval.

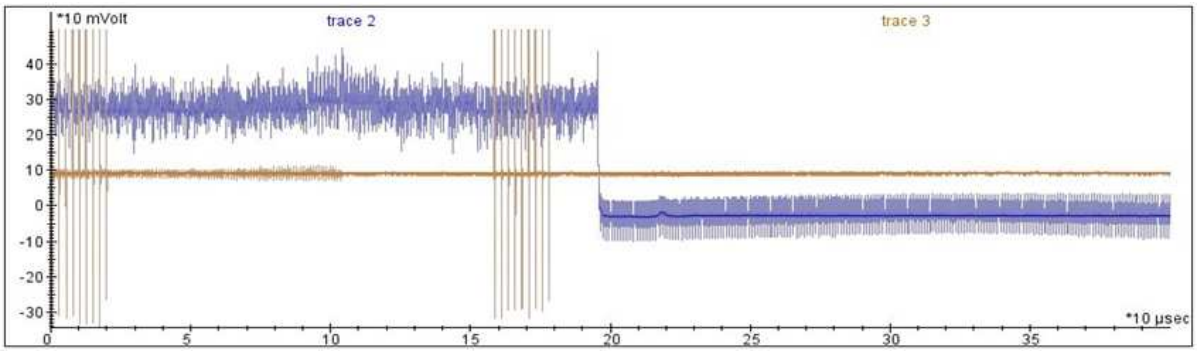

Fig. 2. DES execution and pattern based trigger

Interestingly, we can use this pattern also for a second function: determining correct execution of the card. In earlier experiments we have captured power traces of a card responding to optical faults by terminating itself. In these power traces the synchronization patterns are not present after the DES execution. Hence we can use these patterns to detect, real-time, whether a card is proceeding with normal operation. We program our card reader to switch off power to the card if the second set of patterns does not show up within 100μs of the laser pulse.

With this safeguard set up, we start scanning the chip for a good time/location combination. We find various locations and timings where the card resets or attempts to self-terminate. Within 10000 attempts we find a combination in which the card continues executing properly, but always returns a cipher text that is equal to the plain text. This does not allow obtaining the secret key, but defeats protocols that rely on a user knowing the encrypted result only when in possession of a shared secret key.

We redo the experiments with the 'correct execution' detection switched off. Cards now typically stop responding within 1000 injection attempts. With the detection switched on, we have had one card failure in 60000 injection attempts. When switching off the laser pulse synchronization we get very inconsistent results, as the laser is most of the time not targeting the DES.

These experiments show that on more advanced cards, it is of importance to somehow prevent card termination and to synchronize the pulse with respect to the jitter in the process. In this case this is possible with one pattern, however typically two different patterns are required. This experiment also shows that even without decapping the chip it may be possible to inject faults if the epoxy is transparent.

5.3 Other experimental results

The two cards experimented with before are good for experimentation as they do not sport the full range of modern countermeasures and allow testing aspects of the system in isolation. In this section we would like to share some experimental results on card employing the latest technology.

Backside navigation Navigation for back side attacks is more troublesome than the front side, since no clear image of the chip can be obtained using visible light. Interestingly, a laser pulse usually creates a spike or a dip in the power trace. We scan the surface of the chip, compute an average of the samples that show a glitch effect and put them in a color-based 2D plot, where color coding represents the glitch effect on the power consumption. The presence of different logic and materials in the optical beam path leads to power consumption fluctuations which reveal part of the physical chip layout. Several areas of the chip now can be identified, even though we have no direct access to the front side, as can be seen in 3.

Disabling internal clock Since an internal clock breaks the synchronization with the perturbation environment it is interesting to find a way to disable this clock. As a test case we used a chip that runs on the external clock during start-up, and switches to the internal clock immediately after power up. By experimental insertion of glitches shortly after

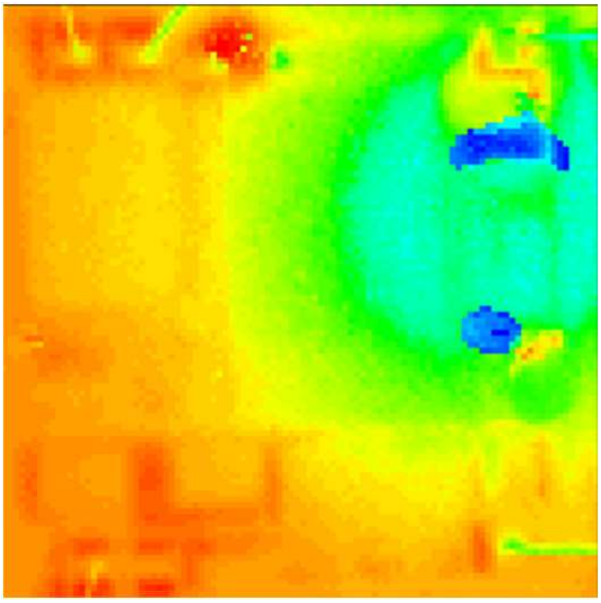

Fig. 3. Backside laser pulse power spike imaging

power up we found the timing offset where the internal clock was activated. To verify whether the switch has been prevented, the frequency spectrum of the power consumption signal can be checked. With a well-chosen glitch we skipped this instruction and prevented switch over to the internal clock. After that we could benefit from the more deterministic chip behavior to identify and exploit other glitching opportunities.

Double verifications We are encountering more cards that, considering the state of the art in double fault injection, verify sensitive operations twice. This implies that three successful injections are necessary: one for the sensitive operation, and two for the verification steps. Under laboratory conditions we have been able to inject a fault in an RSA CRT execution on the latest smart card chip technology. Each verification steps afterwards could be attacked with about 10% success rate using pattern based triggering, resulting in an overall 1% success rate. This means that the expected number of injections needed to retrieve the full private RSA exponent is about 100.

6 Conclusion

In this paper we have detailed the latest developments regarding optical fault injection on secure microcontrollers. A combination of countermeasures makes fault injection less than trivial; however, due to practical security trade-offs made by chip manufacturers and card vendors, optical fault injection is still a possible and affordable method. We developed fault injection methods to challenge the security of the latest smart cards and show experimentally that protected smart cards are still vulnerable to fault injection. Especially the use of an accurate triggering mechanism based on real-time pattern recognition and the use of jitter-free diode lasers shows current countermeasures may be inadequate for the future.

To attain a higher security level it is important for chip manufacturers to decrease the sensitivity of the hardware to optical perturbation, and for software developers to consistently apply a mix of countermeasures. During development it should be presumed single or double glitches will be successful, and therefore we advice using more than two verification steps. As fault injection is a stochastic process, the success rate will drop significantly with each added verification.

Pattern based triggering can be complicated by maximizing the variability of the internal clock frequency and instruction sequence ordering. With the newest fault injection methods we can tune fault injection more carefully, but we cannot fully control the effect of perturbation.

The latest technology may also require injecting faults at two different locations on the chip; e.g. a fault in the cryptographic accelerator and one in the CPU. We expect attacker technology to also move in the direction of two lasers at two locations, even though this will create additional challenges regarding positioning and finding correct parameter locations.

References

Koch99. Paul Kocher, Joshua Jaffe and Benjamin Jun, *"Differential Power Analysis"*, Lecture Notes in Computer Science, Vol. 1666, pp. 388–397, 1999.

Bone01. D. Boneh, R. A. DeMillo, R. Lipton, *"On the Importance of Eliminating Errors in Cryptographic Computations"*, Journal of Cryptology 14(2):101–120, 2001.

Biha97. Eli Biham, Adi Shamir, *"Differential cryptanalysis of the data encryption standard"*, Advances in Cryptology – CRYPTO '97: 17th Annual International Cryptology Conference, LNCS, pp. 513-525, vol. 1294, 1997.

Karp04. Mark Karpovsky, Konrad J. Kulikowski, Er Taubin, *"Differential Fault Analysis Attack Resistant Architectures for the Advanced En-*

cryption Standard", Proc. World Computing Congress, Cardis, pp. 177–192, 2004.

Skor03. S. Skorobogatov and R. Anderson, *"Optical Fault Induction Attacks"*, CHES 2002, LNCS 2523, pp. 2–12, 2003.

Skor10. S. Skorobogatov, *"Optical Fault Masking Attacks"*, FDTC 2010.

Chou06. Hamid Choukri, Michael Tunstall, *"Handbook of information security, Volume 3"*, Hossein Bidgoli (ed.), pp 230–231, John Wiley and Sons, 2006.

Tarn10. C. Tarnovsky, *"Hacking the smartcard chip"*, Blackhat DC 2010.

iso06. *"Identification cards – Integrated circuit cards – Part 3: Cards with contacts – Electrical interface and transmission protocols"*, ISO/IEC 7816-3:2006.

Komm99. O. Kommerling, M Kuhn, *"Design Principles for Tamper-Resistant Smartcard Processors*", USENIX Workshop on Smartcard Technology, Chicago, Illinois, USA, May 10–-11, 1999.

Witt08. M. Witteman., M. Oostdijk, *"Secure Application Programming in the Presence of Side Channel Attacks"*, in RSA conference 2008.

Dero07. O. Derouet, *"Secure Smartcard Design against Laser Fault Injection"*, FTDC 2007.