A Tutorial on Physical Security and Side-Channel Attacks

François Koeune[1,2] and François-Xavier Standaert[1]

[1] UCL Crypto Group
Place du Levant, 3. 1348 Louvain-la-Neuve, Belgium
fstandae@dice.ucl.ac.be – http://www.dice.ucl.ac.be/crypto/
[2] K2Crypt
Place Verte 60 box 2. 1348 Louvain-la-Neuve, Belgium
fkoeune@k2crypt.com – http://www.k2crypt.com/

Abstract. A recent branch of cryptography focuses on the physical constraints that a real-life cryptographic device must face, and attempts to exploit these constraints (running time, power consumption, ...) to expose the device's secrets. This gave birth to implementation-specific attacks, which often turned out to be much more efficient than the best known cryptanalytic attacks against the underlying primitive as an idealized object. This paper aims at providing a tutorial on the subject, overviewing the main kinds of attacks and highlighting their underlying principles.

1 Introduction and objectives

A cryptographic primitive can be considered from two points of view: on the one hand, it can be viewed as an abstract mathematical object or black box (i.e. a transformation, possibly parameterized by a key, turning some input into some output); on the other hand, this primitive will *in fine* have to be implemented in a program that will run on a given processor, in a given environment, and will therefore present specific characteristics.

The first point of view is that of "classical" cryptanalysis; the second one is that of *physical security*. Physical attacks on cryptographic devices take advantage of implementation-specific characteristics to recover the secret parameters involved in the computation. They are therefore much less general – since it is specific to a given implementation – but often much more powerful than classical cryptanalysis, and are considered very seriously by cryptographic devices' implementors.

The goal of this paper is to provide the reader with a first tutorial on physical security. The paper will explore certain of the most important kinds of physical attacks, from direct data probing to electromagnetic analysis. However, the intention is not to make an exhaustive review of existing techniques, but rather to highlight the philosophy underlying the main attacks. So, this is not to be viewed a security manual, but as an introductory course in a specific branch of cryptography.

The authors did their best to keep the paper easy to read, giving a good understanding of the general principle of physical attacks. Strict formalism was sometimes sacrificed to the benefit of intuition, whereas many references were provided to guide the interested reader during his first steps in that fascinating and emerging subject.

Physical attacks usually proceed in two steps: an interaction phase, during which an attacker exploits some physical characteristic of a device (e.g. measures running time or current flow, inserts faults, ...) and an exploitation phase, analyzing this information in order to recover secret information. Although we will discuss the first phase, we will mostly focus on the second: once a "signal" has been obtained, how can we exploit this signal to expose a device's secrets?

1.1 Model

The context of a physical attack is the following: we consider a device capable of performing cryptographic operations (e.g. encryptions, signatures, ...) based on a secret key. This key is stored inside the device, and protected from external access. We assume that an attacker has the device at his disposal, and will be able to run it a number of times, possibly with input values of his choice. In addition, during the device's processing, he will be able to act on or measure some parameters related to the environment, the exact nature of which depends on the attack's context. This can for example be the device's running time, the surrounding electromagnetic field, or some way of inducing errors during the computation. The attacker has of course no direct access to the secret key.

Note that the expression "at disposal" might have various meanings: in some cases, it can be a complete gain of control, like for example by stealing an employee's identification badge during his lunch break, attacking it and then putting it back in place to go unnoticed. As another example, we would like to point out that there are situations where the owner of the device himself might be interested in attacking it, e.g. in the case of a pay-TV decoder chip. On the other hand, the control of the attacker on the device might be much more limited: he could for example be hidden behind the corner of the street when the genuine user is using his device, and monitoring electromagnetic radiations from a distance, or interrogating the device through a web interface, and monitoring the delay between request and answer.

Modern cryptography is driven by the well-known Kerckhoffs' assumption, which basically states that all the secret needed to ensure a system's security must be entirely gathered in the secret keys. In other words, we must assume that an attacker has perfect knowledge of the cryptographic algorithm, implementation details, ... The only thing that he does not know – and which is sufficient to guarantee security – is the value of the secret keys. We will adopt this point of view here, and consider that the attacker is familiar with the device under attack, and that recovering the secret keys is sufficient to allow him to build a pirated device with the same power and privileges as the original one.

1.2 Principle of divide-and-conquer attacks

Most of the attacks we will discuss here are divide-and-conquer attacks. As the name says, divide and conquer attacks attempt to recover a secret key by parts. The idea is to find an observable characteristic that can be correlated with a partial key, small enough to make exhaustive search possible. The characteristic is used to validate the partial key, which can be established independently of the rest of the key. The process is then repeated with a characteristic that can be correlated to another part of the key, and so on until the full key is found, or the remaining unknown part is small enough to be in turn exhaustively searchable.

The word characteristic is intentionally imprecise. Such a characteristic can for example be a power consumption profile during a given time interval, a specific output structure after a fault, a probabilistic distribution of running times for an input subset, ...

Divide and conquer attacks can be iterative (that is, a part of the key must be discovered in order to be able to attack subsequent parts) or not (in which case all parts of the key can be guessed independently). Clearly, an iterative attack will be much more sensible to errors, as a wrong conclusion at some point will make subsequent guesses meaningless. This factor can sometimes be mitigated using an error detection/correction strategy [63].

2 Targets

For the sake of concreteness, principles will be illustrated in two contexts: smart cards (typically representing general purpose microprocessors with a fixed bus length) and FPGAs (typically representing application specific devices with parallel computing opportunities).

2.1 Smart card

One of the most typical targets of side-channel attacks (and one often chosen in the literature) is the smart card. There are several reasons for this. First, these are devices dedicated to performing secure operations. They are also easy to get hold on: usually carried around in many places, small-sized, and an easy target for a pickpocket. In addition, smart cards are pretty easy to scrutinize: as a smart card depends on the reader it is inserted in in many ways (see below), running time, current, ... are easy to monitor. Finally, they are quite simple device, typically running one process at a time, and with a much simpler processor than a desktop PC or mainframe.

Basically, a smart card is a computer embedded in a safe. It consists of a (typically, 8-bit or 32-bit) processor, together with ROM, EEPROM, and a small amount of RAM, which is therefore capable of performing computations. The main goal of a smart card is to allow the execution of cryptographic operations, involving some secret parameter (the key), while not revealing this parameter to the outside world.

Fig. 1. Smart card chip and its connection points: supply voltage (V_{CC}), reset signal (RST), clock (CLK), ground connection (GND), input/output (I/O) and external voltage for programming (V_{PP}, generally not used).

This processor is embedded in a chip and connected to the outside world through eight wires, the role, use, position, ... of which is normalized (Fig. 1). In addition to the input/output wires, the parts we will be the most interested in are the following.

Power supply: smart cards do not have an internal battery. The current they need is provided by the smart card reader. This will make the smart card's power consumption pretty easy to measure for an attacker with a rogue reader.

Clock: similarly, smart cards do not dispose of an internal clock either. The clock ticks must also be provided from the outside world. As a consequence, this will allow the attacker to measure the card's running time with very good precision.

Smart cards are usually equipped with protection mechanisms composed of a shield (the passivation layer), whose goal is to hide the internal behavior of the chip, and possibly sensors that react when the shield is removed, by destroying all sensitive data and preventing the card from functioning properly. This will be discussed further below.

We refer the interested reader to several very good books (e.g. [60]) for a deeper discussion of smart cards.

2.2 FPGA

In opposition to smart cards that are typical standardized general purpose circuits, FPGAs are a good example of circuits allowing application specific implementations. Fundamentally, both smart cards and FPGAs (and most processors and ASICs) share the same leakage sources and are consequently similar in terms of susceptibility against physical attacks. However, it is interesting to consider these two contexts for two reasons:

1. They are generally used for completely different applications: smart cards have limited computational power while FPGAs and ASICs are usually required for their ability to deal with high throughput.
2. FPGAs and ASICs allow parallel computing and have a more flexible architecture (as it is under control of the designer). This may affect their resistance against certain attacks.

More philosophically, we used these two contexts as an illustration (among others) that physical attacks are both general and specific: general because they rely on physical principles that can be applied to any kind of device; specific because when it comes to their practical implementation, their efficiency depends on how well we can adapt these general principles to a particular target.

FPGAs usually contain an array of computational elements whose functionality is determined through multiple programmable configuration bits. These elements, sometimes known as logic blocks, are connected using a set of routing resources that are also programmable (see Figure 2). They can be used to

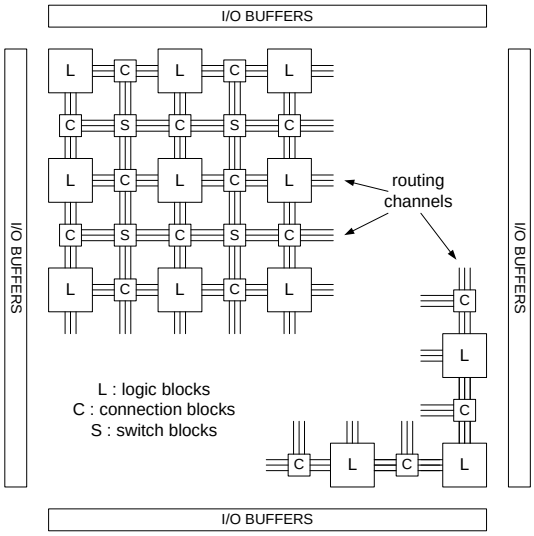

Fig. 2. FPGA: high level view.

implement a variety of digital processing tasks. FPGAs allow the designer to determine the allocation and scheduling of the tasks in the circuit (*e.g.* to trade surface for speed), which is typically out of control of the smart card programmer.

Compared to ASICs, FPGAs present similar design opportunities, although some parts of reconfigurable circuits are not dedicated to the applications but to the reconfigurability management. In brief, FPGAs trade a little bit of the ASICs efficiency for more flexibility. Structural details on FPGAs are not necessary for the understanding of this survey, but can be useful for mounting attacks in practice. A good reference is [23].

2.3 Differences between platforms

Differences between platforms may affect the physical security at two distinct levels. First and practically, the devices may be based on different technologies and consequently have different physical behaviors. For example, certain side-channel attacks require to make predictions of the leakage. The prediction model may be different for different devices. Similarly, fault insertion techniques may have different effects on different technologies. In order to keep our discussions general, we will make abstraction of these possible technological differences and assume in the following that:
(1) Knowing the data handled by a device, it is possible to predict its leakage[3].
(2) Faults can be inserted in the device.

Second, as already mentioned, different platforms may have different architectures, leading to different computational paradigms. For example, smart cards are small processors where the data is managed sequentially. FPGA designs (i.e. the hardware counterpart of smart card programs) have a more flexible architecture and allow parallel computation. Such differences will be emphasized in the following sections.

3 Physical attacks classification

Physical attacks can be classified in many ways. The literature usually sorts them along two orthogonal axes.

Invasive vs. non-invasive: invasive attacks require depackaging the chip to get direct access to its inside components; a typical example of this is the connection of a wire on a data bus to see the data transfers. A non-invasive attack only exploits externally available information (the emission of which is however often unintentional) such as running time, power consumption, ... One can go further along this axis by distinguishing **local** and **distant** attacks: a local attack requires close – but external, i.e. non-invasive – proximity to the device under concern, for example by a direct connection to its power supply. As opposed, a distant attack can operate at a larger distance, for example by measuring electromagnetic field several meters (or hundreds of meters) away, or by interacting with the device through an internet connection.

[3] Remark that most present devices are CMOS-based and their leakages are relatively simple and similar to predict. Typically, models based on the Hamming weight or the Hamming distance of the data processed were successfully applied to target respectively smart cards [50] and FPGAs [67]. On the other hand, technological solutions may also be considered to make the attacks harder. Such possibilities will be discussed in Section 9.

Active vs. passive: active attacks try to tamper with the device's proper functioning; for example, fault-induction attacks will try to induce errors in the computation. As opposed, passive attacks will simply observe the device's behavior during its processing, without disturbing it.

Note that these two axes are well orthogonal: an invasive attack may completely avoid disturbing the device's behavior, and a passive attack may require a preliminary depackaging for the required information to be observable.

These attacks are of course not mutually exclusive: an invasive attack may for example serve as a preliminary step for a non-invasive one, by giving a detailed description of the chip's architecture that helps to find out where to put external probes.

As said in section 2.1, smart cards are usually equipped with protection mechanisms that are supposed to react to invasive attacks (although several invasive attacks are nonetheless capable of defeating these mechanisms). On the other hand, it is worth pointing out that a non-invasive attack is completely undetectable: there is for example no way for a smart card to figure out that its running time is currently being measured. Other countermeasures will therefore be necessary.

The attacks we will consider belong to five major groups.

Probing attacks consist in opening a device in order to directly observe its internal parameters. These are thus invasive, passive attacks.

Fault induction attacks try to influence a device's behavior, in a way that will leak its secrets. The difficulty lies not so much in inducing a fault than in being able to recover secret parameters from the faulty result, and this is the question that will retain most of our attention. These attacks are by essence active, and can be either invasive or non-invasive.

The three last groups are usually denoted as *side-channel attacks*. Their basic idea is to passively observe some physical characteristic during the device's processing, and to use this "side-channel" to derive more information about the processed secret. They are thus passive, and typically non-invasive, although some exceptions exist.

Timing attacks exploit the device's running time.

Power analysis attacks focus on the device's electric consumption.

Electromagnetic analysis attacks measure the electromagnetic field surrounding the device during its processing.

In some sense, timing, power and electromagnetic analysis attacks can be viewed as an evolution in the dimension of the leakage space. Timing attacks exploit a single, scalar information (the running time) for each run. Power attacks provide a one-dimensional view of the device's behavior, namely instant power consumption at each time unit. With the possibility to move the sensor around the

attacked device (or to use several sensors), electromagnetic analysis provide a 4-dimensional view: spatial position and time. This allows for example to separate the contributions of various components of the chip, and therefore to study them separately. Moreover, we will see that EM emanations consist of a multiplicity of signals, which can lead to even more information.

Finally, we believe there is a substantial difference between timing or Simple Power Analysis attacks and subsequent side-channel attacks (this will appear clearly in the next sections): timing attacks and Simple Power Analysis provide an indirect access to the data processed, via the observation of the operations performed. As opposed, power or electromagnetic analysis offer direct access to the data processed.

3.1 About the cost...

From an economical point of view, invasive attacks are usually more expensive to deploy on a large scale, since they require individual processing of each attacked device. In this sense, non-invasive attacks constitute therefore a bigger menace for the smart card industry. According to [66], *"until now, invasive attacks involved a relatively high capital investment for lab equipment plus a moderate investment of effort for each individual chip attacked. Non-invasive attacks require only a moderate capital investment, plus a moderate investment of effort in designing an attack on a particular type of device; thereafter the cost per device attacked is low. [...] semi-invasive attacks can be carried out using very cheap and simple equipment."*

4 Probing

One natural idea when trying to attack a security device is to attempt to depackage it and observe its behavior by branching wires to data buses or observing memory cells with a microscope. These attacks are called probing attacks.

4.1 Measuring phase

The most difficult part of probing attacks lies in managing to penetrate the device and access its internals. An useful tool for this purpose is a *probing station*. Probing stations consist of microscopes with micromanipulators attached for landing fine probes on the surface of the chip. They are widely used in the semiconductor manufacturing industry for manual testing of production-line samples, and can be obtained second-hand for under US$ 10 000.

To make observation easier, the attacker may try to slow down the clock provided to the chip, so that successive states are easily observable. An introduction on probing attacks can be found in [7], and a good overview of ways to depackage a card and probe its content is given in [44].

As we said before, smart cards are usually protected by a passivation layer, which is basically a shield covering the chip, in order to prevent from observing its behavior. In addition, some smart cards are equipped with detectors, for example in the form of additional metallization layers that form a sensor mesh above the actual circuit and that do not carry any critical signals. All paths of this mesh need to be continuously monitored for interruptions and short-circuits, and the smart card has to refuse processing and destroy sensitive data when an alarm occurs. Similarly, monitoring clock frequency and refusing to operate under abnormally low (or high) frequency should be done to protect the chip. Additional sensors (UV, light, ...) may also be placed.

Security is a permanent fight between attackers and countermeasure designers, and these protection means are not invulnerable. According to Anderson [7], "*the appropriate tool to defeat them is the* Focused Ion Beam Workstation *(FIB). This is a device similar to a scanning electron microscope, but it uses a beam of ions instead of electrons. By varying the beam current, it is possible to use it as a microscope or as a milling machine. By introducing a suitable gas, which is broken down by the ion beam, it is possible to lay down either conductors or insulators with a precision of a few tens of nanometers. Given a FIB, it is straightforward to attack a sensor mesh that is not powered up. One simply drills a hole through the mesh to the metal line that carries the desired signal, fills it up with insulator, drills another hole through the center of the insulator, fills it with metal, and plates a contact on top, which is easy to contact with a needle from the probing station*".

Better protection techniques, such as stronger passivation layers, that will make it difficult for the attacker to remove them without damaging the chip itself, are also developed. They complicate the attacker's task, but do not make it impossible yet. An interesting example, discussing how such a stronger passivation layer was defeated, can be found in [56].

4.2 Exploitation phase

The most obvious target is of course the part of memory where secret keys are stored; similarly, in a software-based device, the attacker can also tape the data buses connecting memory to processor, as he knows that the secret key will of course be processed during the signature (or decryption), and hence transit through that wire. From our pedagogical point of view, this kind of attack is not extremely interesting: being able to access smart card internals might be a strong technical challenge (which is out of our scope), but exploiting this information is straightforward.

Things might get more difficult (and interesting) when only part of the information can be read (for example because technical constraints allow only to tape a part of the data bus, providing two bits of each transferred word), or when countermeasures are at stake, for example bus scrambling, which can be

thought as some sort of lightweight encryption used for internal transfer and storage. However, we will not discuss these topics more in the detail here. We refer the interested reader to [31, 27, 32] for further information.

5 Fault induction attacks

When an electronic device stops working correctly, the most natural reaction is to get rid of it. This apparently insignificant habit may have deep impact in cryptography, where faulty computations are sometimes the easiest way to discover a secret key.

As a matter of fact, a recent and powerful cryptanalysis technique consists in tampering with a device in order to have it perform some erroneous operations, hoping that the result of that erroneous behavior will leak information about the secret parameters involved. This is the field of fault induction attacks.

5.1 Types of faults

The faults can be characterized from several aspects.

Permanent vs. transient: as the name says, a permanent fault damages the cryptographic device in a permanent way, so that it will behave incorrectly in all future computations; such damage includes freezing a memory cell to a constant value, cutting a data bus wire, ... As opposed, with a transient fault, the device is disturbed during its processing, so that it will only perform fault(s) during that specific computation; examples of such disturbances are radioactive bombing, abnormally high or low clock frequency, abnormal voltage in power supply, ...

Error location: some attacks require the ability to induce the fault in a very specific location (memory cell); others allow much more flexibility;

Time of occurrence: similarly, some attacks require to be able to induce the fault at a specific time during the computation, while others do not;

Error type: many types of error may be considered, for example:

- flip the value of some bit or some byte,
- permanently freeze a memory cell to 0 or 1,
- induce (with some probability) flips in memory, but only in one direction (e.g. a bit can be flipped from 1 to 0, but not the opposite),
- prevent a jump from being executed,
- disable instruction decoder,
- ...

As can be guessed, the fault model has much importance regarding the feasibility of an attack. In fact, two types of papers can be found in the literature: the first type deals with the way to induce errors of a given type in current devices; the second basically assumes a (more or less realistic) fault model and deals with the way this model can be exploited to break a cryptosystem, without bothering

with the way such faults can be induced in practice. These two types are of course complementary to determine the realism of a new attack and the potential weaknesses induced by a new fault induction method. From the viewpoint we took in this tutorial, we are mostly interested in the second aspect, i.e. how we can exploit faulty computations to recover secret parameters. However, let us first briefly consider the other aspect: fault induction methods.

5.2 Fault induction techniques

Faults are induced by acting on the device's environment and putting it in abnormal conditions. Many channels are available to the attacker. Let us review some of them.

Voltage: Unappropriate voltage might of course affect a device's behavior. For example, smart card voltages are defined by ISO standards: a smart card must be able to tolerate on the contact VCC a supply voltage between $4,5$V and $5,5$V, where the standard voltage is specified at 5V. Within this range the smart card must be able to work properly. However, a deviation of the external power supply, called spike, of much more than the specified 10% tolerance might cause problems for a proper functionality of the smart card. Indeed, it will most probably lead to a wrong computation result, provided that the smart card is still able to finish its computation completely.

Clock: Similarly, standards define a reference clock frequency and a tolerance around which a smart card must keep working correctly. Applying an abnormally high or low frequency may of course induce errors in the processing. Blömer and Seifert [10] note that "*a finely tuned clock glitch is able to completely change a CPU's execution behavior including the omitting of instructions during the executions of programs*". Note that, as opposed to the clock slowing down described in section 4, whose goal was to make internal state easier to observe, this clock variation may be very brief, in order to induce a single faulty instruction or to try to fool clock change detectors.

Temperature: Having the device process in extreme temperature conditions is also a potential way to induce faults, although it does not seem to be a frequent choice in nowadays attacks.

Radiations: Folklore often presents fault induction attacks as "microwave attacks" (the attacker puts the smart card into a microwave oven to have it perform erroneous computations). Although this is oversimplified, it is clear that correctly focused radiations can harm the device's behavior.

Light: Recently, Skorobogatov and Anderson [66] observed that illumination of a transistor causes it to conduct, thereby inducing a transient fault. By applying an intense light source (produced using a photoflash lamp magnified with a microscope), they were able to change individual bit values in an SRAM. By the same technique, they could also interfere with jump instructions, causing conditional branches to be taken wrongly.

Eddy current: Quisquater and Samyde [56] showed that eddy currents induced by the magnetic field produced by an alternating current in a coil could induce various effects inside a chip as for example inducing a fault in a memory cell, being RAM, EPROM, EEPROM or Flash (they could for example change the value of a pin code in a mobile phone card).

Several papers and books address the issue of fault induction techniques. We refer the reader to [7, 5, 6, 29, 30, 46] and, for the last two techniques, to [66] and [56].

5.3 Cryptanalyses based on fault

Attack on RSA with CRT Fault induction attack on RSA[4] with Chinese Remaindering Theorem (CRT) [12, 37] is probably the most exemplary instance of fault induction attack: first, it is very easy to explain, even to a non-cryptologist; second, it is also easy to deploy, since only one fault induction *somewhere* in the computation – even with no precise knowledge of that fault's position – is enough to have it work; third, it is extremely powerful, as having one faulty computation performed is sufficient to completely break a signature device.

Implementations of RSA exponentiation often make use of the Chinese Remaindering Theorem to improve performance. Let m be the message to sign, $n = pq$ the secret modulus, d and e the secret and public exponents. Exponentiation process is described in Alg. 1. Of course several values involved in this algorithm are constant and need not be recomputed every time.

Algorithm 1 Chinese Remaindering Theorem

$m_p = m \bmod p$
$m_q = m \bmod q$
$d_p = d \bmod (p-1)$
$d_q = d \bmod (q-1)$
$x_p = m_p^{d_p} \bmod p$
$x_q = m_q^{d_q} \bmod q$
$s = \text{chinese}(x_p, x_q) = q(q^{-1} \bmod p)x_p + p(p^{-1} \bmod q)x_q \bmod n$
return s

Suppose an error occurs during the computation of either x_p or x_q (say x_p, to fix ideas, and denote by x'_p the incorrect result)[5]. It is easy to see that, with overwhelming probability, the faulty signature s' derived from x'_p and x_q will be such that

[4] A short description of RSA can be found in Appendix A.

[5] Note that, since these two computations are by far the most complex part of the full signature process, inducing a transient fault at random time during the computation has great chance to actually affect one of these.

$$s'^e \equiv m \bmod q,$$
$$s'^e \not\equiv m \bmod p.$$

Consequently, computing

$$\gcd(s'^e - m \bmod n, n)$$

will give the secret factor q with overwhelming probability. With this factorization of n, it is straightforward to recover the secret exponent d from e. As we see, having the cryptographic device perform one single faulty signature (without even the need to compare it to correct computations) is sufficient to be able to forge any number of signatures. Moreover, the type of fault is very general, and should therefore be fairly easy to induce.

Differential fault analysis Shortly after the appearing of the fault attack against RSA, Biham and Shamir [9] showed that such attacks could be applied against block ciphers as well, by introducing the concept of *differential fault analysis* (DFA).

They demonstrated their attack against the Data Encryption Standard[6]. The fault model they assume is that of transient faults in registers, with some small probability of occurrence for each bit, so that during each encryption/decryption there appears a small number of faults (typically one) during the computation, and that each such fault inverts the value of one of the bits[7].

The basic principle of their attack is that of a divide and conquer attack: suppose that we dispose of two results, one correct and one faulty, for the same input value. Suppose further that a (unique) fault has occurred, and that that fault affected one bit of R_{15} (that is, the right part of the input to the last round of DES – see Fig. 3(a), with $i = 15$). This bit will follow two paths through this last round. First, R_{15} will be copied into L_{16}, which will also have exactly one wrong bit. Second, it will be part of the input to one[8] S-box (Fig. 3(b)) and, being faulty, will induce a 4-bit error in this S-box output. This in turn will affect 4 bits of R_{16}. Noting that all further operations (IP^{-1}, P, ...) are deterministic and do not depend on unknown values, we can, from the final outputs (the right and the wrong) of DES, identify which bit of R_{15} was incorrect. We

[6] A description of DES can be found in Appendix B.

[7] The authors claim that their model is the same as that of [12, 37] but, in our opinion, this claim is misleading: whereas RSA's fault induction works provided *any* error occurs during the computation, DES's DFA requires that only one (or a very small number of) bit(s) is (are) affected by the error. This model is therefore much less general.

[8] In fact, due to the expansion function, a single bit could affect two S-boxes. The argument is the same in this case, and we omit it for simplicity.

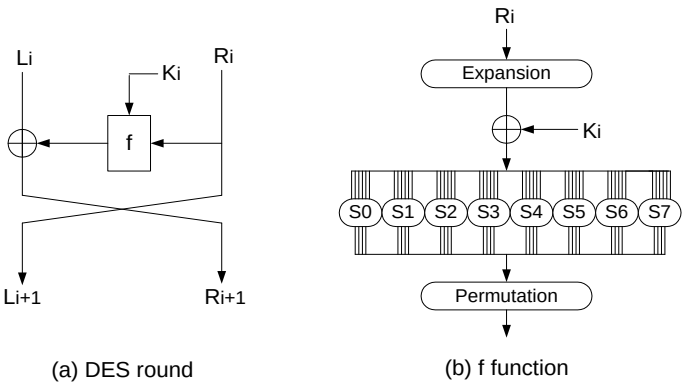

Fig. 3. Data Encryption Standard.

can also trace back the error as close as possible to the involved S-box. Taking the exclusive-or of the two outputs, we end up with a relationship of the form

$$S(R \oplus K) \oplus S(R \oplus F \oplus K) = X,$$

where S denotes the S-box under concern, R the part of interest of R_{15} (that we can reconstruct from the output), K the corresponding part of the key, and F the one-bit fault. Note that we do not know the S-box output, nor, of course, the key. The non-linearity of the S-box will help us: as a matter of fact, not all input pairs could have produced this output difference. So, we will simply perform an exhaustive search over all possible (that is, 2^6) key values, and discard all values which do not yield the expected difference. According to Biham and Shamir, only four possibilities remain on the average. Repeating the experiment, it is possible to confirm the correct 6-bit value, and then to attack other key parts.

To induce an error with the expected form and location, the attacker will repeat encryptions with device put under extreme conditions, and with same plaintext as input, until he observes a difference between ciphertexts with the expected pattern (one wrong bit in the output corresponding of L_{16} and four wrong bits in R_{16}). Choosing the time at which error is triggered to target the last rounds will of course be helpful.

Similar arguments can be used if the fault occurred in rounds 14 or 15. Using this technique, Biham and Shamir could recover a full DES key using between 50 and 200 messages. Note that triple-DES can be attacked in the same way.

It is important to remark at this point that fault induction significantly depends on the target device. While, due to the simplicity of the processor, it may be relatively easy to insert of fault during a specified computation in a smart card, large parallel designs (e.g. block ciphers implemented on FPGAs) may be more challenging.

Other results Others fault models have also been considered, which allow pretty trivial attacks. Some authors, for example, consider a model in which memory cells can be flipped from one to zero (or from zero to one), but not the opposite. An obvious way to exploit this is to repeatedly induce faults on the key, until all its bits have been forced to zero (and producing some ciphertexts between each fault induction). The chain is then explored backwards, starting from the known (null) key, and guessing at each step which bits have been flipped; correct guesses are identified by comparison with the ciphertexts. An even simpler attack is that of [10], that additionally assumes that it is possible to choose the location of the flipped bit. In this case, the attack simply consists in forcing a key bit to zero and checking if the result is different from the one obtained without fault induction. If this is the case, conclude the key bit was 1, otherwise conclude 0.

Finally, several obvious ways to exploit very specific faults can easily be devised: for example, a fault that would affect a loop counter so that only two or three rounds of DES are executed would of course allow to break the scheme. Similarly, disabling the instruction decoder could have the effect that all instructions act as a NOP so the program counter cycles through the whole memory.

6 Timing attack

6.1 Introduction

Usually the running time of a program is merely considered as a constraint, some parameter that must be reduced as much as possible by the programmer. More surprising is the fact that the running time of a cryptographic device can also constitute an *information channel*, providing the attacker with invaluable information on the secret parameters involved. This is the idea of *timing attack*. This idea was first introduced by Kocher [42].

In a timing attack, the information at the disposal of the attacker is a set of messages that have been processed by the cryptographic device and, for each of them, the corresponding running time. The attacker's goal is to recover the secret parameters (fig. 4). Remember that, as was said in section 2.1, the clock ticks are provided to the smart card by the terminal. Precise timing measurements are therefore easy to obtain.

6.2 Timing attack against RSA with Montgomery reduction

The ideas suggested by Kocher were first practically applied by Dhem et al. against the RSA algorithm, implemented using Montgomery multiplication [25]. Although this attack is not optimal, its idea is pretty intuitive, and fits our tutorial purpose, so we will sketch it in this section.

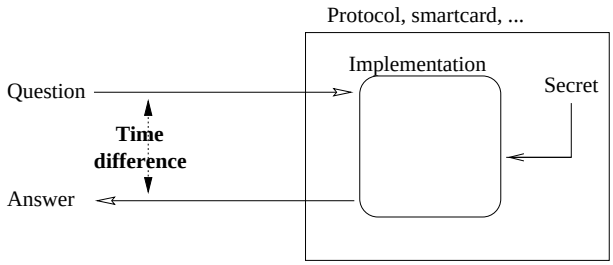

Fig. 4. The timing attack principle.

The context is that of an RSA signature, and the goal of the attacker is to recover the secret exponent d. A common method to perform a modular exponentiation is the square and multiply algorithm (Alg. 2 [9]). This algorithm is mainly a sequence of modular multiplications and squares (which we will view as simple multiplications of a value by itself). When implemented in a scholar way, modular multiplications are time-consuming operations. Montgomery [52] proposed a clever way to speed-up these operations, by transferring them to a modulus which is better suited to the machine's internal structure.

Algorithm 2 Square and multiply

$x = m$
for $i = \omega - 2$ downto 0 **do**
  $x = x^2 \bmod n$
  if $d_i == 1$ **then**
    $x = x \cdot m \bmod n$
  end if
end for
return x

For simplicity, we will not describe Montgomery's algorithm in the detail here. For our purpose, it is sufficient to know that, for fixed modulus, the time for a Montgomery multiplication is constant, independently of the factors, except that, if the intermediary result of the multiplication is greater than the modulus, an additional subtraction (called a reduction) has to be performed. In other words, this means that, depending on the input, the running time of a Montgomery multiplication can take two values, a "short one" (no final reduction) or a "long one" (final reduction needed). Of course, for given input values, we can predict whether the multiplication will be long or short.

The attack is an iterative divide and conquer attack: we will start by attacking the first unknown bit $d_{\omega-2}$, and, at each step of the attack, assume we know bits $d_{\omega-1} \dots d_{i+1}$ and recover bit d_i.

[9] Here, $d_{\omega-1}$ denotes the most significant bit of d (which we assume to be equal to 1) and d_0 denotes the lsb.

Let us begin by $d_{\omega-2}$. If this bit is equal to 1, the first loop pass in Alg 2 will involve a multiplication by m (line 5). As we have seen, this multiplication can be either long or short and, since no secret value is involved before this step, we can, for a given message m, determine whether this multiplication would be long or short. What we will do is partition our set of messages according to this criterion: all messages for which that first multiplication would be long will be put in subset A, and all messages for which that multiplication will be short will be put in subset B.

What is the influence of this step on the total running time? For all messages of subset A, the first pass in the loop has taken slightly more time than for all messages of subset B. What we expect is that this first pass will on average have a noticeable influence on the total running time, so that actual total running times for messages from subset A will be slightly longer. We cannot simulate further passes in the loop, since their behavior (and hence the evolution of x) depend on secret parameters $b_{\omega-3} \dots b_0$. So we will simply consider them as noise, hoping that their influence on messages from subset A will globally be the same than on subset B.

An important point in the attack is that the simulation and partition are based on the assumption that the first bit of the secret exponent is equal to 1. If this is not the case, then the conditional step we are targeting is not executed. Thus, our predictions of long or short operations will not correspond to any reality[10]. Our partition in sets A and B should then be roughly random, and there is no reason why we should observe any difference between the running times of subset A and B.

To conduct the attack, we will simply revert that argument: assuming $d_{\omega-2} = 1$, we build a partition as described above, and compare the average running times of the two subsets. If the running times are significantly different, we conclude that our assumption that $d_{\omega-2} = 1$ was true, otherwise, we conclude $d_{\omega-2} = 0$.

Once this bit has been determined, we have enough information to simulate the square and multiply until the second pass, and hence attack the next bit. Note that this does not require a new measurement set: we simply build a new partition of the same set.

[10] This argument is a bit too simplistic: in fact, the successive Montgomery multiplications of a square and multiply are not independent. Characterizing this fact allowed to greatly improve the attack's efficiency, but this falls outside the scope of this tutorial.

6.3 General idea

The above argument can be generalized as follows. For a given algorithm involving a secret parameter, we view the global running time as a sum of random variables $T = \sum_{i=1}^{N} T_i$ corresponding to the various steps of the algorithm, and each individual execution of the algorithm as a realization of this random variable.

If we can – based on a key guess – simulate the computation and its associated running time up to step k, we can filter the measured running times by subtracting the parts corresponding to these steps. So, if input value m_j yielded a measured running time t_j and a simulated computation time t_j^{EST}, we estimate the remaining running time as $t_j^{REM} = t_j - t_j^{EST}$, corresponding to a realization of the random variable $T^k = \sum_{i=k+1}^{N} T_i$. If our key guess is correct, this filtering should reduce the variance (and the correct guess corresponds to the one that reduces variance the most).

We can generalize this further by characterizing the probabilities for an observation under the admissible hypotheses and the a priori distribution of these hypotheses, and deriving a decision strategy. The appropriate statistical tools for this purpose is maximum likelihood estimation or, better, optimal decision strategy. Schindler [62, 63] applied an optimal decision strategy to the above scenario (square and multiply with Montgomery multiplication), and showed that this led to a drastic improvement in the attack's efficiency.

7 Power analysis attacks

In addition to its running time, the power consumption of a cryptographic device may provide much information about the operations that take place and the involved parameters. This is the idea of power analysis, first introduced by Kocher et al. in [43], that we describe in the context of a smart card implementation.

7.1 Measuring phase

Measuring the power consumption of a smart card is a pretty easy task: as the clock ticks, the card's energy is also provided by the terminal, and can therefore easily be obtained. Basically, to measure a circuit's power consumption, a small (e.g., 50 ohm) resistor is inserted in series with the power or ground input. The voltage difference across the resistor divided by the resistance yields the current. Well-equipped electronics labs have equipment that can digitally sample voltage differences at extraordinarily high rates (over 1GHz) with excellent accuracy (less than 1% error). Devices capable of sampling at 20MHz or faster and transferring the data to a PC can be bought for less than US$ 400.

7.2 Simple power analysis

Simple Power Analysis (SPA) attempts to interpret the power consumption of a device and deduce information about the performed operations or involved parameters. This is better illustrated by an example. Fig. 5, taken from the original description of power analysis, shows the consumption curve (named a *trace*) of a device performing a DES operation. It clearly shows a pattern that is repeated 16 times and corresponds to the 16 rounds of DES. The lower part of the figure is a detailed view of two rounds, providing more information about the round's substeps.

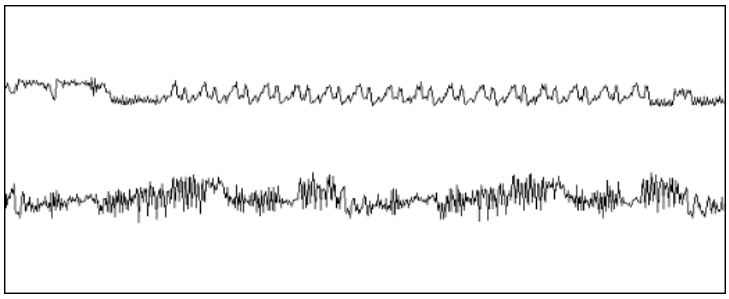

Fig. 5. SPA monitoring from a single DES operation performed by a typical smart card [43]. The upper trace shows the entire encryption operation, including the initial permutation, the 16 DES rounds, and the final permutation. The lower trace is a detailed view of the second and third rounds.

Of course, this information is not an attack in itself. Everybody knows that DES has 16 rounds, and knowing that a device is performing a DES encryption does not expose its secrets at all. According to our Kerckhoffs assumption, the cryptographic algorithm is known to the attacker anyway. However, there are cases in which this sequence of operations can provide useful information, mainly when the instruction flow depends on the data. For example, let us come back to exponentiation through the square and multiply algorithm. If the square operation is implemented differently than the multiply – a tempting choice, as this will allow specific optimizations for the square operation, resulting in faster code – and provided this difference results in different consumption patterns, then the power trace of an exponentiator directly yields the exponent's value. Similarly, some hardware multipliers behave differently when one of their operands is zero, which allows immediate return of the result, but exposes this input value. Generally speaking, all programs involving conditional branch operations depending on secret parameters are at risk.

However, power consumption may also depend on the *data* manipulated. This leads to a more general class of attacks that is investigated in the next section. So, SPA typically targets variable instruction flow, whereas Differential Power Analysis and its variants target data-dependence.

In practice, instruction flow exposing is a point where the security of smart cards and FPGAs may differ. On the one hand, sequential computing in smart cards involves that at one specific instant, only one of the operations is running, e.g. either square or multiply in our previous example. This makes it possible to distinguish operations by a simple visual inspection of the power traces. On the other hand, in FPGAs, parallel computation (if used in the design) prevents this visual inspection, as the power consumption will not only be the one of the targeted operation.

7.3 Differential Power Analysis

As we said, the idea of a divide and conquer attack is to compare some characteristic to the values being manipulated. One of the simplest comparison tools we can imagine is mean comparison, used in the original power analysis attack: Differential Power Analysis (DPA) [43]. As it is pretty intuitive, we will begin by studying it, in the context of DES encryption.

Here, the characteristic we will focus on is an arbitrary output bit b of an arbitrary S-box at the 16th round (say the first bit of S_1's output), and we will observe this characteristic through its associated power consumption. In fact, we have no idea of when this value is actually computed by the cryptographic device, nor do we know what its associated power consumption may look like. The only thing we know for sure is that, at some point in time, that value will be manipulated (computed, stored, ...) by the cryptographic device and that the power consumption of the device depends on this data. To expose that value, we will use a method very similar to the one we used for timing attack: partitioning and averaging.

Let us first summarize the idea. We will perform a large number of encryptions with variable inputs and record the corresponding consumption curves. Then, we will partition our set in two subsets: one, A, for which the target bit was equal to 0, the other, B for which the target bit was 1. In order to do this partitioning, we need to be able to estimate the value of bit b. We have access to the value R_{15} that entered the last round function, since it is equal to L_{16} (Fig. 3(a)). Looking at the round function in more detail (Fig. 3(b)), we see that six specific[11] bits of R_{15} will be XORed with six specific bits of the key, before entering the S-box. So, doing a guess on a 6-bit key value, we are able to predict the value of b. This means that we actually have 2^6 partitions of our measurements, one for each possible key.

We will then compute the average power consumption for messages in subset A, and subtract it from the average power consumption for messages in subset B (for each possible partition). Since all computations for subset A involved

[11] By specific we mean that they are chosen according to fixed permutation functions, so we can identify them independently of any key value.

manipulating a 0 at a point where all computations for subset B involved manipulating a 1, we should observe notable differences at points in time where this value was actually manipulated, whereas we expect other points, which are unrelated to b, to behave as noise and be cancelled in subtraction. Of course, this will only be true provided our initial key guess is correct, and we will in fact use our observation as an oracle to validate this key guess: only the correct key guess (corresponding to the correct partition) should lead to significant differences.

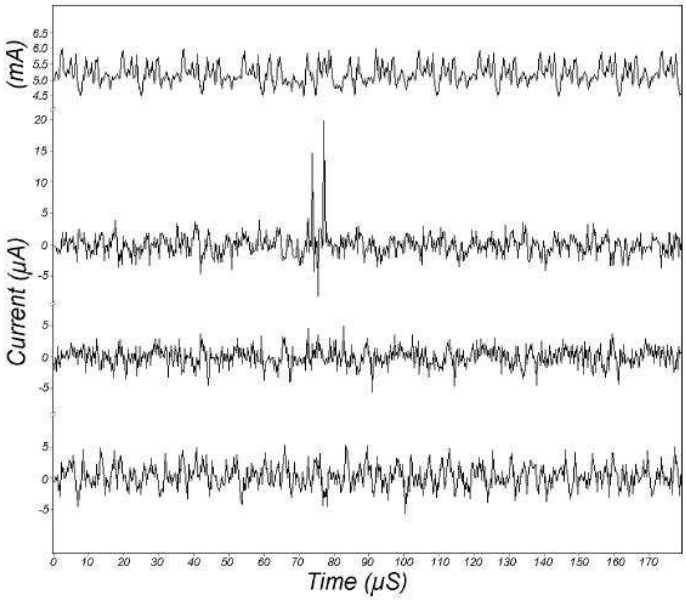

Fig. 6. DPA traces from [43]

Figure 6, taken from the original paper describing DPA, shows traces corresponding to correct and incorrect key guesses: the top trace is the average power consumption among the whole sample; the curve below shows the differential trace corresponding to the right key guess (and hence right partition), and the two bottom curves show differential traces for wrong key guesses. The spikes are clearly visible.

According to Messerges, such a difference of mean test is relevant because, at some point during a software DES implementation, the microprocessor needs to compute b. When this occurs or any time data containing this selection bit is manipulated, there will be a slight difference in the amount of power dissipated, depending on the values of these bits. What we just did is in fact building a *model* of the smart card behavior. The model is, in this case, pretty elementary, but this modelling phase is nonetheless an indispensable part of the attack.

Finally, once this 6-bit subkey has been determined, the attack goes on by focusing on other S-boxes and target bits.

7.4 Correlation attack

Looking at the previous descriptions, it is easy to see that a DPA is far from making an optimal use of the sampled measurements. In this section, we describe two possible improvements.

A first improvement, usually denoted as the "multiple bit" DPA, comes when observing that the key guess performed in a DPA does not only allow to predict the bit b, but all the four output bits of S_1. As a consequence, one may separate the measurements in two sets according to these multiple bit values: one set corresponding to "*all zeroes*", the other one to "*all ones*". This improves the attack signal to noise ratio. However, as multiple bit attacks only consider the texts that give rise to "*all something*" values, it is suboptimal and a lot of texts (measurements) are actually not used. The correlation analysis allows solving this problem and constitutes a second improvement. Correlation Power Analysis usually hold in three steps.

- First, the attacker *predicts* the power consumption of the running device, at one specific instant, in function of certain secret key bits. For example, let us assume that the power consumption of the DES implementation depends of the Hamming weight of the data processed[12]. Then, the attacker could easily predict the value of S_1's output Hamming weight, for the 2^6 possible values of the key guess and N different input texts. This gives 2^6 possible predictions of the device power consumption and the result of this prediction is stored in a **prediction matrix**.
- Secondly, the attacker *measures* the power consumption of the running device, at the specific time where it processes the same input texts as during the prediction phase. The result of this measurement is stored in a **consumption vector**.
- Finally, the attacker *compares* the different predictions with the real, measured power consumption, using the correlation coefficient[13]. That is, he computes the correlation between the consumption vector and all the columns of the prediction matrix (corresponding to all the 2^6 key guesses). If the attack is successful, it is expected that only one value, corresponding to the correct key guess, leads to a high correlation coefficient.

[12] As already mentioned, this is a typical model for the power consumed in smart cards.

[13] Let $M(i)$ denote the ith measurement data (*i.e.* the ith trace) and M the set of traces. Let $P(i)$ denote the prediction of the model for the ith trace and P the set of such predictions. Then we calculate:

$$C(M,P) = \frac{\mu(M \times P) - \mu(M) \times \mu(P)}{\sqrt{\sigma^2(M) \times \sigma^2(P)}} \tag{1}$$

where $\mu(M)$ denotes the mean of the set of traces M and $\sigma^2(M)$ its variance.

Such an attack has been successfully applied to a variety of algorithms and implementations, e.g. in [13, 55, 67]. As an illustration, Figure 7 shows the result

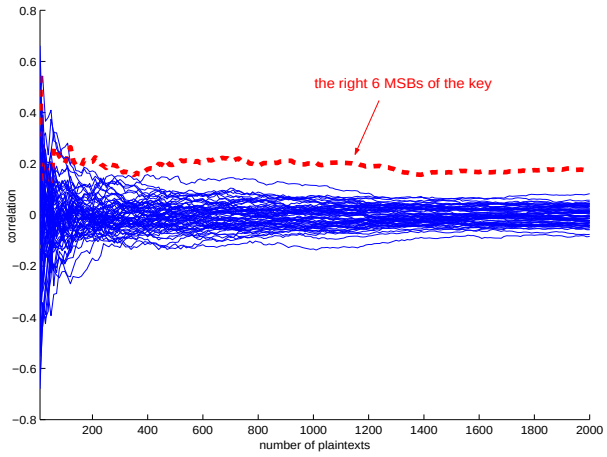

Fig. 7. A correlation attack against the Data Encryption Standard.

of a correlation attack against an implementation of the DES [68]. It is clearly observed that the correct key is distinguishable after 500 measurements.

To conclude this section, a number of points are important to notice:

- Correlation power analysis (as any power analysis attack targeting data dependencies) requires a power consumption model. The quality of this model has a strong influence on the attack efficiency. However, this leakage model does not mandatorily have to be known a priori by the attacker and can be built from measurements. This is the context of template attacks.
- Next to the leakage model, the synchronization of measurements is another typical issue in side-channel attacks. Fundamentally, the easiest synchronization is to perform the statistical test on the complete sampled data (this is typically done in DPA). However, when the statistical test becomes computationally intensive, such a technique may become cumbersome to deal with in practice. Some knowledge about the design or more advanced synchronization techniques (e.g. FFT-based) usually solve the problem.
- As SPA, DPA-like attacks are affected by parallel computing. This is easily seen when observing the DES design in Appendix B. Say we observe an 8-bit smart card implementation. When S_1's output is computed, the sampled power consumption relates to the four predicted S-box output bits and 4 other (unknown) bits. These unknown bits basically add some algorithmic noise to the measurements. Say we observe an FPGA implementation with all the S-boxes computed in parallel. Then, only 4 bits are predicted out of the

32 actually computed in the device. This means that the power consumption will be affected by more algorithmic noise. In general, as already mentioned in the context of fault insertion attacks, parallel computing improves security against certain physical attacks.
– Remark finally that the described correlation power analysis is still not optimal. Techniques based on a maximum likelihood detection of the correct key offer even better results. However, as the power consumption models in use (i.e. based on the Hamming weight, distance of the data processed) have a linear dependence, correlation provides a simple and efficient tool for attacking devices in practice.

8 EMA

Any movement of electric charges is accompanied by an electromagnetic field. *Electromagnetic attacks*, first introduced by Quisquater and Samyde [58], and further developed in [59, 26] exploit this side channel by placing coils in the neighborhood of the chip and studying the measured electromagnetic field.

The information measured can be analyzed in the same way as power consumption (simple and differential electromagnetic analysis – SEMA and DEMA – or more advanced correlation attacks), but may also provide much more information and are therefore very useful, even when power consumption is available[14]. As a matter of fact, 3D positioning of coils might allow to obtain much more information from the device's components. Moreover, Agrawal et al. [2] showed that EM emanations consist of a multiplicity of signals, each leaking somewhat different information about the underlying computation. They sort the EM emanations in two main categories: direct emanations, i.e. emanations that result from intentional current flow, and unintentional emanations, caused by coupling effects between components in close proximity. According to them, unintentional emanations can prove much more useful that direct emanations. Moreover, some of them have substantially better propagation than direct emanations, which enables them to be observed without resorting to invasive attacks (and even, in some cases, to be carried out at pretty large distances - 15 feet! - which brings it to the field of tempest-like attacks [1]).

To summarize, EMA measurement phase is much more flexible and challenging that power measurement phase, and the provided information offers a wide spectrum of potential information. On the other hand, this information may be exploited using the same basic or advanced techniques as for power analysis, even if optimal decision models can be adapted to the specificities of EMA.

[14] One can of course imagine contexts in which power consumption cannot be obtained, but where it is possible to measure the radiated field, for example from a short distance.

In essence, EMA is a non-invasive attack, as it consists in measuring the near field. However, this attack is made much more efficient by depackaging the chip first, to allow nearer measurements and to avoid perturbations due to the passivation layer.

9 Countermeasures

Countermeasures against physical attacks range among a large variety of solutions. However, in the present state of the art, no single technique allows to provide perfect security, even considering a particular attack only. Protecting implementations against physical attacks consequently intends to make the attacks harder. In this context, the implementation cost of a countermeasure is of primary importance and must be evaluated with respect to the additional security obtained. The exhaustive list of all possible solutions to protect cryptographic implementations from physical opponents would deserve a long survey in itself. In this section, we will only suggest a few exemplary proposals in order to illustrate that security can be added at different levels. We refer the reader to [57] for a more comprehensive (although slightly outdated) review of existing countermeasures.

The most direct way to prevent physical opponents is obviously to act at the physical/hardware level. A typical example related to probing is the addition of shields, conforming glues, or any process that makes the physical tempering of a device more complex. Detectors that react to any abnormal circuit behaviors (light detectors, supply voltage detectors, ...) may also be used to prevent probing and fault attacks [69]. With respect to side-channel attacks, simple examples of hardware protection are noise addition, or the use of detachable power supplies [65]. However, as detailed in [22, 47], none of these methods provide a perfect solution.

Close to the physical level, technological solutions can also be considered. There exist circuit technologies that offer inherently better resistance against fault attacks (e.g. dual rail logic [72]) or side-channel attacks (e.g. any dynamic and differential logic style [45, 70]). While these solutions may be very interesting in practice, because they can be combined with good performance, they do not offer any theoretical guarantee of security either. For example, dynamic and differential logic styles usually make the modelling of the power consumption more difficult (no simple prediction, smaller data dependencies of the leakage) but theoretically, such a model can always be obtained by the mean of template attacks (see next section).

Finally, most of the proposed techniques aim to counteract fault and side-channel attacks at the algorithmic level. Certain solutions, such as the randomization of the clock cycles, use of random process interrupts [48] or bus and memory encryption [14, 27], may be used to increase the difficulty of successfully attacking

a device, whatever physical attack is concerned. Most solutions however relate to one particular attack. With respect to faults, they mainly include different kinds of error detection/correction methods based on the addition of space or time redundancies [64, 41, 35, 40]. The issue regarding these techniques is that their cost is frequently close to naive solutions such as duplication or repetition. Regarding side-channel attacks, a lot of countermeasures intend to hide (or mask) the leakage or to make it unpredictable. One could for example think about performing a "square and multiply always" algorithm (i.e. always perform a multiplication and discard the result if the operation was not necessary) for exponentiation, and using a similar "reduce always" method for Montgomery multiplication in order to counter timing attacks[15]. Another countermeasure consists in preventing the attacker from being able to make predictions about intermediary values, a simple example of which is Kocher's blinding [42]. Other typical examples are in [4, 19, 28]. Once again, such protections are not perfect and, in the case of block ciphers, for example, have been shown to be susceptible to higher-order side-channel attacks. Moreover, their implementation cost is also an issue.

In general, good security may nevertheless be obtained by a sound combination of these countermeasures. However, it is more an engineering process than a scientific derivation of provably secure schemes. Theoretical security against physical attacks is a large scope for further research.

10 More advanced attacks and further readings

Most advanced scenarios generally aim to improve the attacks' efficiency (i.e. to reduce the number of physical interactions required), to make them more general or to defeat certain countermeasures. We give here typical examples of such improvements that are also good directions for further readings.

For power and Electromagnetic analysis: (1) Multi-channel attacks [3] basically aim to improve the attacks' efficiency by making an optimal use of different available leakages, based on a maximum likelihood approach. (2) Template attacks [20] intend to remove the need for a leakage model and consequently make power and electromagnetic analysis more general. For this purpose, they act in two separate steps: (a) build a model based on sampled data, using a template of the target device (b) use maximum likelihood to determine the key whose leakage profile matches the best the one of the device under attack. Remark that templates inherently allow to defeat a large number of countermeasures. (3) Second-order attacks [51, 71] finally target implementations in which the leakage has been masked. Contrary to first order attacks that aim to predict the power consumption of one specific operation at one specific instant during a cryptographic computations, higher-order techniques take advantage of correlations existing between multiple operations, possibly performed at different

[15] These are of course naive methods, which have been widely improved in the literature.

instants.
Another research lead is to explore the feasibility of an attack in a different context. For example, some authors demonstrated that timing attacks can also be conducted against a classical desktop computer, or even against a server accessed through a network connection [15, 8]. Moreover, Bernstein argues that the complexity of a general-purpose CPU (multiple cache levels, pipelining, ...) might make timing attacks extremely difficult to prevent on such platforms.

Elliptic curve cryptography has been the field of a large number of publications regarding physical attacks. As elliptic curve operations are basically transpositions of classical operations in another mathematical structure, many of the aforementioned attacks can be transposed to this context too. On the other hand, this structure offers a degree of flexibility that allows several specific countermeasures to be designed in an efficient way (see [36, 34, 33, 24, 21], to only name a few).

11 Conclusion

In this paper, we presented a number of practical concerns for the security of cryptographic devices, from an intuitive point of view. It led us to cover recently proposed attacks and discuss their respective characteristics as well as possible countermeasures. The discussion clearly underlines that probing, side-channel or fault attacks constitute a very significant threat for actual designers. The generality of these attacks was suggested in the tutorial, as we mainly investigated algorithmic issues. However, it must be noted that the actual implementation of physical attacks (and more specifically, their efficiency) may be significantly platform-dependent. Moreover, the presented techniques usually require a certain level of practical knowledge, somewhat hidden in our abstract descriptions.

From an operational point of view, the present state of the art suggests that good actual security may be obtained by the sound combination of countermeasures. However, significant progresses are still required both in the practical setups for physical interactions with actual devices and in the theoretical understanding of the underlying phenomenons. The development of a theory of provable security is therefore a long term goal.

12 Acknowledgements

The authors are grateful to W. Schindler for useful comments.

References

1. *NSA tempest series*, Available at `http://cryptome.org/#NSA--TS`.
2. D. Agrawal, B. Archambeault, J.R. Rao, and P. Rohatgi, *The EM side channel*, in Kaliski et al. [38].
3. D. Agrawal, J. R. Rao, and P. Rohatgi, *Multi-channel attacks*, Proceedings of the 5th International Workshop on Cryptographic Hardware and Embedded Systems (CHES) (Cologne, Germany) (C. Walter, Ç. K. Koç, and C. Paar, eds.), Lecture Notes in Computer Science, vol. 2779, Springer-Verlag, September 7-10 2003, pp. 2–16.
4. M.-L. Akkar and C. Giraud, *An implementation of DES and AES, secure against some attacks*, in Çetin K. Koç et al. [16].
5. R. Anderson and M. Kuhn, *Tamper resistance – a cautionary note*, Proc. of the second USENIX workshop on electronic commerce (Oakland, California), Nov. 18-21 1996, pp. 1–11.
6. _____, *Low cost attacks on tamper resistant devices*, Proc. of 1997 Security Protocols Workshop, Lectures Notes in Computer Science (LNCS), vol. 1361, Springer, 1997, pp. 125–136.
7. R.J. Anderson, *Security engineering*, Wiley & Sons, New York, 2001.
8. Daniel J. Bernstein, *Cache-timing attacks on AES*, Available at `http://cr.yp.to/antiforgery/cachetiming-20050414.pdf`, November 2004.
9. E. Biham and A. Shamir, *Differential fault analysis of secret key cryptosystems*, Proc. of Advances in Cryptology – Crypto '97 (Berlin) (Burt Kaliski, ed.), vol. 1294, Springer-Verlag, 1997, Lecture Notes in Computer Science Volume 1294, pp. 513–525.
10. J. Blömer and J.P. Seifert, *Fault based cryptanalysis of the advanced encryption standard (AES)*, Cryptology ePrint Archive: Report 2002/075. Available at `http://eprint.iacr.org`.
11. D. Boneh (ed.), *Advances in cryptology - CRYPTO '03*, Lectures Notes in Computer Science (LNCS), vol. 2729, Springer-Verlag, 2003.
12. D. Boneh, R.A. DeMillo, and R.J. Lipton, *On the importance of checking cryptographic protocols for faults*, Advances in Cryptology - EUROCRYPT '97, Konstanz, Germany (W. Fumy, ed.), LNCS, vol. 1233, Springer, 1997, pp. 37–51.
13. E. Brier, C. Clavier, and F. Olivier, *Correlation power analysis with a leakage model*, proceedings of CHES, Lectures Notes in Computer Science (LNCS), vol. 3156, Springer, 2004, pp. 16–29.
14. E. Brier, H. Handschuh, and C. Tymen, *Fast primitives for internal data scrambling in tamper resistant hardware*, in Çetin K. Koç et al. [16], pp. 16–27.
15. B. Canvel, A. Hiltgen, S. Vaudenay, and M. Vuagnoux, *Password interception in a SSL/TLS channel*, in Boneh [11].
16. Çetin K. Koç, David Naccache, and Christof Paar (eds.), *Cryptographic Hardware and Embedded Systems - CHES 2001*, Lectures Notes in Computer Science (LNCS), vol. 2162, Springer-Verlag, August 2001.
17. Çetin K. Koç and Christof Paar (eds.), *Cryptographic Hardware and Embedded Systems - CHES '99*, Lectures Notes in Computer Science (LNCS), vol. 1717, Springer-Verlag, August 1999.
18. Çetin K. Koç and Christof Paar (eds.), *Cryptographic Hardware and Embedded Systems - CHES 2000*, Lectures Notes in Computer Science (LNCS), vol. 1965, Springer-Verlag, August 2000.

19. S. Chari, C. Jutla, J. Rao, and P. Rohatgi, *Towards sound approaches to counteract power-analysis attacks*, Advances in Cryptology - CRYPTO '99 (M. Wiener, ed.), Lectures Notes in Computer Science (LNCS), vol. 1666, Springer-Verlag, 1999.
20. S. Chari, J.R. Rao, and P. Rohatgi, *Template attacks*, in Kaliski et al. [38].
21. Benoît Chevallier-Mames, Mathieu Ciet, and Marc Joye, *Low-cost solutions for preventing simple side-channel analysis: Side-channel atomicity*, IEEE Transactions on Computers **53** (2004), no. 6.
22. C. Clavier, J.S. Coron, and N. Dabbous, *Differential power analysis in the presence of hardware countermeasures*, in Kaliski et al. [39].
23. K. Compton and S. Hauck, *Reconfigurable computing: A survey of systems and software*, ACM Computing Surveys **34** (2002), no. 2.
24. J.-S. Coron, *Resistance against differential power analysis for elliptic curves cryptosystems*, in Çetin K. Koç and Paar [17].
25. J.-F. Dhem, F. Koeune, P.-A. Leroux, P. Mestré, J.-J. Quisquater, and J.-L. Willems, *A practical implementation of the timing attack*, Proc. CARDIS 1998, Smart Card Research and Advanced Applications (J.-J. Quisquater and B. Schneier, eds.), LNCS, Springer, 1998.
26. K. Gandolfi, C. Mourtel, and F. Olivier, *Electromagnetic analysis: Concrete results*, Proc. of Cryptographic Hardware and Embedded Systems (CHES 2001) (Çetin Kaya Koç, David Naccache, and Christof Paar, eds.), Lecture Notes in Computer Science, vol. 2162, Springer, 2001, pp. 251–261.
27. Jovan D. Golic, *DeKaRT: A new paradigm for key-dependent reversible circuits*, in Kaliski et al. [39], pp. 98–112.
28. L. Goubin and J. Patarin, *DES and differential power analysis: the duplication method*, in Çetin K. Koç and Paar [17].
29. P. Gutmann, *Secure deletion of data from magnetic and solid-state memory*, Proc. of 6th USENIX Security Symposium, 1997, pp. 77–89.
30. ———, *Data remanence in semiconductor devices*, Proc. of 7th USENIX Security Symposium, 1998.
31. Helena Handschuh, Pascal Paillier, and Jacques Stern, *Probing attacks on tamper-resistant devices*, in Çetin K. Koç and Paar [17].
32. Yuval Ishai, Amit Sahai, and David Wagner, *Private circuits: Securing hardware against probing attacks*, in Boneh [11].
33. K. Itoh, J. Yajima, M. Takenaka, and N. Torii, *DPA countermeasures by improving the window method*, in Kaliski et al. [38].
34. T. Izu and T. Takagi, *Fast parallel elliptic curve multiplications resistant to side channel attacks*, Proc. of PKC '2002 (David Naccache and Pascal Paillier, eds.), Lecture Notes in Computer Science, vol. 2274, Springer, 2002, pp. 335–345.
35. N. Joshi, K. Wu, and R. Karry, *Concurrent error detection schemes for involution ciphers*, in Çetin K. Koç and Paar [18], pp. 400–412.
36. M. Joye and J.-J. Quisquater, *Hessian elliptic curves and side-channel attacks*, in Çetin K. Koç et al. [16].
37. Marc Joye, Arjen K. Lenstra, and Jean-Jacques Quisquater, *Chinese remaindering based cryptosystems in the presence of faults*, Journal of cryptology **12** (1999), no. 4, 241–245.
38. Burton S. Kaliski, Çetin K. Koç, and Christof Paar (eds.), *Cryptographic Hardware and Embedded Systems - CHES 2002*, Lectures Notes in Computer Science (LNCS), vol. 2523, Springer-Verlag, August 2002.
39. Burton S. Kaliski, Çetin K. Koç, and Christof Paar (eds.), *Cryptographic Hardware and Embedded Systems - CHES 2003*, Lectures Notes in Computer Science (LNCS), vol. 2779, Springer-Verlag, September 2003.

40. M. Karpovsky, K.J. Kulikowski, and A. Taubin, *Differential fault analysis attack resistant architectures for the advanced encryption standard*, proceedings of CARDIS 2004.
41. R. Karri, G. Kuznetsov, and M. Gössel, *Parity-based concurrent error detection of substitution-permutation network block ciphers*, in Kaliski et al. [39].
42. P. Kocher, *Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems*, Advances in Cryptology - CRYPTO '96, Santa Barbara, California (N. Koblitz, ed.), LNCS, vol. 1109, Springer, 1996, pp. 104–113.
43. P. Kocher, Jaffe J., and B. Jub, *Differential power analysis*, Proc. of Advances in Cryptology – CRYPTO '99 (M. Wiener, ed.), LNCS, vol. 1666, Springer-Verlag, 1999, pp. 388–397.
44. Olivier Kömmerling and Markus G. Kuhn, *Design principles for tamper-resistant smartcard processors*, Proc. of USENIX Workshop on Smartcard Technology (Smartcard '99), 1999.
45. F. Mace, F.-X. Standaert, I. Hassoune, J.-D. Legat, and J.-J. Quisquater, *A dynamic current mode logic to counteract power analysis attacks*, proceedings of DCIS, 2004.
46. D.P. Maher, *Fault induction attacks, tamper resistance, and hostile reverse engineering in perspective*, Financial Cryptography: First International Conference (FC '97) (R. Hirschfeld, ed.), Lectures Notes in Computer Science (LNCS), vol. 1318, Springer-Verlag, 1997.
47. S. Mangard, *Hardware countermeasures against DPA - a statistical analysis of their effectiveness*, proceedings of CT-RSA, Lecture Notes in Computer Science, vol. 2964, Springer-Verlag, 2004, pp. 222–235.
48. D. May, H. Muller, and N. Smart, *Randomized register renaming to foil DPA*, in Çetin K. Koç et al. [16].
49. A.J. Menezes, P.C. van Oorschot, and S.A. Vanstone, *Handbook of applied cryptography*, CRC Press, 1997.
50. T. S. Messerges, E. A. Dabbish, and R. H. Sloan, *Investigations of power analysis attacks on smartcards*, Proc. USENIX Workshop on Smartcard Technology, 1999.
51. Th.S. Messerges, *Using second-order power analysis to attack DPA resistant software*, in Çetin K. Koç and Paar [18].
52. P.L. Montgomery, *Modular multiplication without trial division*, Mathematics of Computation **44** (1985), no. 170, 519–521.
53. National Bureau of Standards, *FIPS 197, Advanced Encryption Standard*, Federal Information Processing Standard, NIST, U.S. Dept. of Commerce,November 2001.
54. ———, *FIPS PUB 46, The Data Encryption Standard*, Federal Information Processing Standard, NIST, U.S. Dept. of Commerce, Jan 1977.
55. S.B. Ors, F. Gurkaynak, E. Oswald, and B. Preneel, *Power-analysis attack on an asic aes implementation*, proceedings of ITCC, 2004.
56. J.-J. Quisquater and D. Samyde, *Eddy current for magnetic analysis with active sensor*, Proc. of Esmart 2002, 2002.
57. Jean-Jacques Quisquater and François Koeune, *Side-channel attacks: state-of-the-art*, CRYPTREC project deliverable, available at `http://www.ipa.go.jp/security/enc/CRYPTREC/fy15/doc/1047_Side_Channel_report.pdf`, October 2002.
58. Jean-Jacques Quisquater and David Samyde, *A new tool for non-intrusive analysis of smart cards based on electro-magnetic emissions: the SEMA and DEMA methods*, Eurocrypt rump session, 2000.

59. ———, *Electromagnetic analysis (EMA): measures and countermeasures for smart cards*, Smart cards programming and security (e-Smart 2001), Lectures Notes in Computer Science (LNCS), vol. 2140, Springer, 2001, pp. 200–210.
60. W. Rankl and W. Effing, *Smart card handbook*, John Wiley & Sons, 1997.
61. R. Rivest, A. Shamir, and L. Adleman, *A method for obtaining digital signatures and public key cryptosystems*, **21** (1978).
62. W. Schindler, *Optimized timing attacks against public key cryptosystems*, Statistics & Decisions (2000), to appear.
63. W. Schindler, J.-J. Quisquater, and F. Koeune, *Improving divide and conquer attacks against cryptosystems by better error detection correction strategies*, Proc. of 8th IMA International Conference on Cryptography and Coding (Berlin) (B. Honary, ed.), Springer, December 2001, Lecture Notes in Computer Science Volume 2260, pp. 245–267.
64. A. Shamir, *How to check modular exponentiation*, Presented at the rump session of EUROCRYPT '97, Konstanz, Germany.
65. A. Shamir, *Protecting smart cards from passive power analysis with detached power supplies*, in Çetin K. Koç and Paar [18].
66. S. Skorobogatov and R. Anderson, *Optical fault induction attacks*, in Kaliski et al. [38].
67. F.-X. Standaert, S.B. Ors, and B. Preneel, *Power analysis of an fpga implementation of rijndael: is pipelining a dpa countermeasure?*, proceedings of CHES, Lectures Notes in Computer Science (LNCS), vol. 3156, Springer, 2004, pp. 30–44.
68. F.-X. Standaert, S.B. Ors, J.-J. Quisquater, and B. Preneel, *Power analysis attacks against FPGA implementations of the DES*, proceedings of FPL, Lecture Notes in Computer Science, vol. 3203, Springer-Verlag, 2004, pp. 84–94.
69. H. Bar-El *et al.*, *The sorcerer's apprentice guide to fault attacks*, Tech. Report 2004/100, IACR eprint archive, 2004, Available at http://eprint.iacr.org.
70. K. Tiri, M. Akmal, and I. Verbauwhede, *A dynamic and differential CMOS logic with signal independent power consumption to withstand differential power analysis on smart cards*, proceedings of ESSCIRC, 2003.
71. J. Waddle and D. Wagner, *Towards efficient second-order power analysis*, proceedings of CHES, Lecture Notes in Computer Science, vol. 3156, Springer-Verlag, 2004, pp. 1–15.
72. J.D. Waddle and D.A. Wagner, *Fault attacks on dual-rail encoded systems*, Tech report UCB//CSD-04-1347, UC Berkeley, August 23, 2004.

A RSA

RSA, named after the initials of its authors, Rivest, Shamir and Adleman [61] is probably the most famous asymmetric encryption (and signature) primitive. It basically goes as follows:

1. Alice chooses two large prime numbers p and q and computes their product $n = pq$ and $\phi(n) = (p-1)(q-1)$.
2. She also chooses a value e that has no common factor with $\phi(n)$ and computes $d = e^{-1} \bmod \phi(n)$.
3. Alice publishes (n, e) as her public key, and keeps d as her private key.
4. To send her a message m (with $0 \leq m < n$), Bob computes $c = m^e \bmod n$.
5. Alice decrypts c by computing $c^d \bmod n$. By Euler's theorem, it can easily be shown that the result is equal to m.

We refer the reader to [49] for more information on RSA.

B The Data Encryption Standard : a case study

In 1977, the DES algorithm [54] was adopted as a Federal Information Processing Standard (FIPS) for unclassified government communication. Although a new Advanced Encryption Standard was selected in October 2000 [53], DES is still widely used, particularly in the financial sector. DES encrypts 64-bit blocks with a 56-bit key and processes data with permutations, substitutions and XOR operations. It is a good example of Feistel cipher and its structure allows very efficient hardware implementations.

Basically, the plaintext is first permuted by a fixed permutation *IP*. Next the result is split into two 32-bit halves, denoted with L (left) and R (right) to which a round function is applied 16 times. The ciphertext is calculated by applying the inverse of the initial permutation *IP* to the result of the 16th round.

The secret key is expanded by the key schedule algorithm to sixteen 48-bit round keys K_i and in each round, a 48-bit round key is XORed to the text. The key schedule consists of known bit permutations and shift operations. As a consequence, finding any round key bit directly involves that the secret key is corrupted.

The round function is represented in Figure 8 (a) and is easily described by:

$$\begin{aligned} L_{i+1} &= R_i \\ R_{i+1} &= L_i \oplus f(R_i, K_i) \end{aligned}$$

where f is a nonlinear function detailed in Figure 8 (b): the R_i part is first expanded to 48 bits with the E box, by doubling some R_i bits. Then, it performs a bitwise modulo 2 sum of the expanded R_i part and the 48-bit round key K_i. The

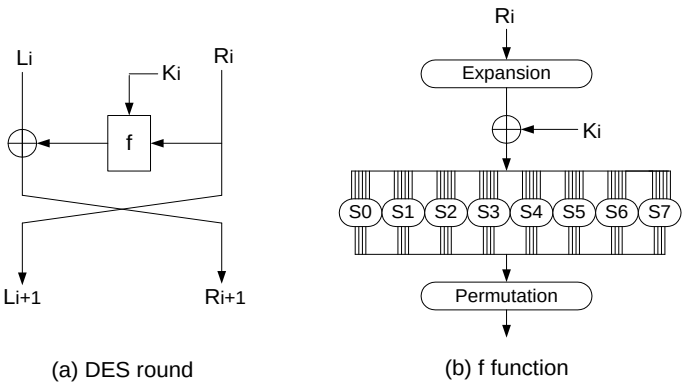

Fig. 8. Data Encryption Standard.

output of the XOR function is sent to eight non-linear S-boxes. Each of them has six input bits and four output bits. The resulting 32 bits are permuted by the bit permutation P.

Finally, DES decryption consists of the encryption algorithm with the same round keys but in reversed order.