Side channel attacks

State-of-the-art

Evaluator: Prof. Jean-Jacques Quisquater, Math RiZK, consulting

Scientific Support: Dr. François Koeune, K2Crypt

Executive summary.

This report describes the current state-of-the-art about side channel cryptanalysis. It will describe the various side channels known in the literature and discuss them from various points of view (conditions of application, ease of deployment, relative efficiency, ...) The question of countermeasures will also be explored. We will review the (large number of) countermeasures proposed in the literature, compare them and attempt to distinguish the most efficient ones.

Our conclusion is that, at the moment, no perfect protection exists. By using appropriate countermeasures, it is possible to make the attacker's task harder, but not to make it impossible yet. One must therefore start by defining the adversary the device must resist again, and the resources he disposes of, before choosing appropriate countermeasures against this adversary.

Although we tried to keep the various countermeasures in perspective, the large number of problems to be taken into account, and the highly application-dependent character of side channel attacks make it impossible to advise adequate countermeasure in a general framework. This topic needs therefore to be further studied in each practical case.

Finally, we insist on the quickly evolving character of this topic, and therefore recommend to keep informed on the evolution of the field.

Contents

1 Preliminary remark

Probably the first target of side channel cryptanalysis are tamper-resistant devices, the best example of which are smart cards. For the sake of concreteness, the discussion below will often be put in that context, although most of it applies to other cryptographic devices as well. Specific matter, such as the protection of pure hardware implementations, will also be considered.

2 Introduction

A cryptographic primitive can be considered from two points of view: on the one hand, it can be viewed as an abstract mathematical object (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 *side-channel cryptanalysis*. Side-channel cryptanalysis takes advantage of implementation-specific characteristics to recover the secret parameters involved in the computation. It is therefore much less general – since it is specific to a given implementation – but often much more powerful than classical cryptanalysis, and is considered very seriously by cryptographic devices' implementors.

This report describes the current state-of-the-art about side channel cryptanalysis. It will describe the various side channels known in the literature and discuss them from various points of view (conditions of application, ease of deployment, relative efficiency, ...). Section 3 gives a brief introduction to the smart card, and the components of interest for the side channel attacker. Section 4 gives a classification of the various attacks, depending on the way they affect the attacked device. Sections 5 to 9 describe the side channel attacks known so far, namely probing attacks, fault induction attacks, timing attacks, single and differential power analysis and electromagnetic attacks. Section 10 then reviews each of these attacks from a (both hardware and software) countermeasures' point of view, reviewing the (large number of) countermeasures proposed in the literature, comparing them and attempting to distinguish the most efficient ones. The section finishes by reviewing elliptic curve-specific issues.

3 Smart card overview

This section will very briefly introduce the concept of a smart card. We refer the interested reader to several very good books (e.g. [75]) for a deeper introduction to the subject.

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. As opposed, the goal of the attacker is to recover this secret parameter.

This processor is embedded in a chip and connected to the outside world through eight wires, the role, use, position, ... of which is normalized. 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 the attacker.

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 behaviour of the chip and possibly sensors that react when the shield is removed, by destroying all sensitive data and preventing the card to function properly. This will be discussed further below.

4 Classification of side channel attacks

The literature usually classifies side channel attacks along two orthogonal axes.

Invasive vs. non-invasive: invasive attacks require depackaging the chip to get direct access to its 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,... In [80], Skorobogatov and Anderson add a new distinction with what they call *semi-invasive attacks*. These attacks have the specificity that they require depackaging of the chip to get access to the chip surface, but do not tamper with the passivation layer – they do not require electrical contact to the metal surface.

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

Note that these two axes are well orthogonal: an invasive attack may completely avoid disturbing the card's behaviour, 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 3, smart cards are usually equipped with protection mechanisms that are supposed to react to invasive attacks (although several invasive attacks are nonetheless capable to defeat these mechanisms, as will be illustrated below). 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.

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 [80], *"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."*

5 Probing attacks

The first way to attack a smart card is to depackage it and observe its behaviour by branching wires to the data bus or observing memory cells with a microscope. This task can be made easier using 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 [5], and a very good overview of ways to depackage a card and probe its content is given in [57].

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 behaviour. This layer has to be removed first, but this is usually not a problem for the attacker [57, 72]. 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.

Unfortunately, these protection means are not invulnerable. According to Anderson [5], "*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 [72].

6 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 behaviour will leak information about the secret parameters involved. This is the field of fault induction attacks.

6.1 Types of faults

The fault 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, etc; as opposed, with a transient fault, the device is disturbed during its processing, so that it will perform fault(s) in that specific computation; examples of such disturbances are radioactive bombing, abnormally high or low clock frequency, abnormal voltage in power supply, etc.

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,
- ...

Here too, some attacks do not care with the type of error that has to occur.

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. This report will first deal with attacks in a given fault model, and then with fault induction techniques.

6.2 Cryptanalyses based on fault

6.2.1 Attack on RSA with CRT

Fault induction attack on RSA with Chinese Remaindering Theorem (CRT) [15, 48] 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. Moreover, the widespread use of RSA contributes to making this attack much more than a simple theoretical threat. In fact, a significant number of papers (even very recent ones) deal with this specific attack and ways to counter it.

For all these reasons, we believe it to be worth describing the attack's principle (a full discussion on the attack, as well as variants against other targets, can be found in [48, 15]).

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)[1]. It is easy to see that, with overwhelming probability, the faulty signature s' derived from x'_p and x_q will be such that

$$\begin{aligned} s'^e &\equiv m \bmod q, \\ s'^e &\not\equiv m \bmod p. \end{aligned}$$

Consequently, computing

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

will give the secret factor q. 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.

6.2.2 Differential fault analysis

In [11], Biham and Shamir propose a fault attack, called *differential fault analysis* (DFA), and more oriented towards symmetric encryption schemes.

[1] 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.

They demonstrate their attack against DES. 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[2].

The basic principle of their attack is the following: the attacker encrypts some (possibly unknown) plaintext twice, once without fault induction, and once with fault induction (in practice, if the fault induction is probabilistic, he repeats encryptions until he observes a difference between ciphertexts). If the fault occurred in the right half of round 16, then only one bit in the right half of the ciphertext (before the final permutation) differs between the two ciphertexts. The left half of the ciphertext can differ only in output bits of the S-box (or two S-boxes) to which this single bit entered. In such a case, we can guess the six key bits of each such S-box in the last round, and discard any value which disagrees with the expected difference. On the average, only four possible 6-bit values of the key remain. Similar arguments can be used if the fault occured in rounds 14 or 15. Using this technique, they could recover a full DES key using between 50 and 200 messages. Note that triple-DES can be attacked in the same way.

In the same paper, Biham and Shamir also propose techniques to identify the keys of completely unknown ciphers and to reconstruct their complete specifications, provided they have a DES-like structure.

Finally, they develop another attack against DES in a permanent fault model (e.g. by cutting a wire or destroying a memory cell), which is sometimes considered more realistic.

6.2.3 Attacks on elliptic curve cryptography

DFA was later adapted to elliptic curve cryptosystems [10]. The fault model is the same as above, i.e. they assume the possibility to flip a single bit in a register during the computation. The attack basically works by modifying one of the parameters defining the underlying curve, with the effect that the computation is transferred onto another curve, hopefully weaker than the original one. Other fault models are also explored.

[2]The authors claim that their model is the same as that of [15, 48] 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.

6.2.4 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 [14], 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.

Other papers exist in the literature, that propose specific fault induction attacks, but, to our knowledge, none of them presents significantly new ideas compared to those exposed in previous sections. Of course this does not mean that no other algorithms can be attacked: [11] notes for example that DFA can break block ciphers like IDEA, RC5, Blowfish, and many additional ones; with some modifications, stream ciphers can be attacked as well; similarly, [48] shows that the attack against RSA can be applied to many variants such as LUC, KMOV, ...

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.

In short, we could summarize this by saying that, if any type of fault can be induced, then any cryptosystem can trivially be broken. This brings us to the next question: "which fault inductions are possible?"

6.3 Fault induction techniques

Faults are induced in a smart card by acting on its environment and putting it in abnormal conditions. Many channels are available to the attacker. Let us review some of them.

Voltage: As required by ISO standards, a smart card IC 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 smartcard IC. Indeed, it will most probably lead to a wrong computation result, provided that the smart card IC is still able to finish its computation completely.

Clock: Similarly, standards define a reference clock frequency and a tolerance around which the card must keep working correctly. Applying an abnormally high or low frequency may of course induce errors in the processing. Blömer and Seifert [14] 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 5, 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 card 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 card's behaviour.

Light: Recently, Skorobogatov and Anderson [80] 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: Recently too, Quisquater and Samyde [72] 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 [5, 6, 7, 35, 36, 59] and, for the last two techniques, to [80] and [72].

7 Timing attacks

7.1 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 [54], and then practically implemented against an RSA implementation using the Montgomery algorithm by a team involving the authors of the present report [27]. Several other papers were devoted to improvements of this attack and to timing attacks against other targets.

7.2 The model

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. His goal is to recover the secret parameters (fig. 1).

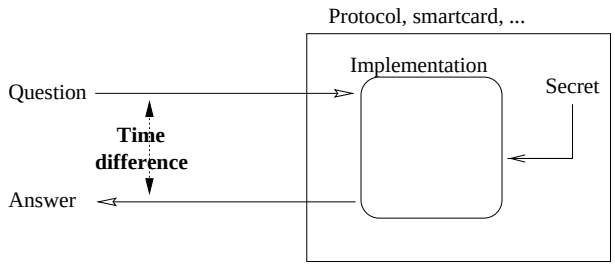

Figure 1: The timing attack principle.

Remember that, as was said in section 3, the clock ticks are provided to the smart card by the terminal. Precise timing measurements are therefore easy to obtain.

7.3 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 [27]. This section will describe the basic principle of this attack[3].

Consider an algorithm A that, given a message m as input, performs some computation involving a secret parameter (the key) k. We note:
M, the set of messages,
K, the set of keys,
S, the output set,
$A : M \times K \rightarrow S : (m, k) \rightarrow A(m, k)$, the computation with input m and key k,
$T : M \times K \rightarrow \mathbb{R} : m \rightarrow t = T(m, k)$, the time taken to compute $A(m, k)$.
$O : M \rightarrow \{0, 1\} : m \rightarrow O(m)$, an oracle, based on our knowledge of the implementation, that provides us with some information about the details of the computation of $A(m, k)$.

Remark: It may look surprising that the oracle does not depend on the key k, although the computation of $A(m, k)$ does, but this is precisely the idea of the attack: typically, we want to build a decision criterion (formalized by the oracle) that will be meaningful or not, *depending on the actual value of some bit*[4] *of the key*. By observing the meaningfulness of our criterion, we will deduce the bit value.

The scenario of our attack is the following: Eve disposes of a set of messages and, for each of them, the time needed to compute $A(m, k)$. Her goal is to recover the parameter k, which is supposed to be constant throughout the attack. To simplify our notations, we will thus simply note $T(m)$ instead of $T(m, k)$.

To attack bit i of the key k, Eve will use an oracle O to build two subsets of messages $M_1, M_2 \subseteq M$. We will denote the corresponding timings by the functions:

$$F_1 : M_1 \rightarrow \mathbb{R} : m \rightarrow F_1(m) = T(m)$$

$$F_2 : M_2 \rightarrow \mathbb{R} : m \rightarrow F_2(m) = T(m)$$

[3]Note that the attack of [27] is in fact a bit more complex and more efficient than the one described below. We will however stick to the basic principle.

[4]To simplify notations, we will consider here that the attack is bit-oriented, i.e. tries to recover the key on a bit-by-bit basis.

Suppose these two functions have the following properties:

$$\begin{cases} \text{If } k_i = 0, \text{ then} & F_1 \text{ is a random variable } v_1^0 \\ & F_2 \text{ is a random variable } v_2^0 \\ \text{If } k_i = 1, \text{ then} & F_1 \text{ is a random variable } v_1^1 \\ & F_2 \text{ is a random variable } v_2^1 \end{cases}$$

and suppose that these variables follow different distributions. By observing the distributions of the actual samples, and matching them to the random variables they are the closest to, it should be possible to deduce the value of k_i.

Suppose for example that, for some function ϕ:

$$\phi(v_1^0) = \phi(v_2^0) \text{ and } \phi(v_1^1) > \phi(v_2^1)$$

then with the following statistical test:

$$H_0 : \phi(F_1) \stackrel{?}{=} \phi(F_2)$$
$$H_1 : \phi(F_1) \stackrel{?}{>} \phi(F_2)$$

we deduce that if H_0 is accepted with error probability α, then $i = 1$ with error probability α.

To summarize, Eve is able to construct two (or more) sets of messages, and functions whose statistical behaviours will depend on the actual value of the bit i. By observing the relative behaviours of the functions, Eve will be able to determine, with a certain error probability, the value of the bit i.

7.3.1 Square and multiply algorithm and Montgomery multiplication

To fix notations, consider the computation of $m^k \bmod n$, where the secret exponent k has binary notation $(k_{\omega-1}, k_{\omega-2}, \ldots, k_0)$, and $k_{\omega-1}$ denotes the most significant bit. The algorithm is the left to right square and multiply (alg. 2). Both multiplication and square are assumed to be done using Montgomery's algorithm.

When implemented in a scholar way, modular multiplications are time-consuming operations. Montgomery [64] 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 = ω − 2 downto 0 do
  x = x^2 mod n
  if k_i == 1 then
    x = x · m mod 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.

7.3.2 The timing attack

The most obvious way to take advantage of this knowledge is to aim our attack at the *multiply* step of the square and multiply. The idea is the following:

We start by attacking $k_{\omega-2}$, the second bit[5] of the secret key. Performing the square and multiply algorithm step-by-step, we see that, if that bit is 1, then the value $m \cdot m^2$ will have to be computed during the square and multiply.

Now, for some messages m (those for which the intermediary result of the multiplication will be greater than the modulus), an additional reduction will have to be performed during this multiplication, while, for other messages, that reduction step will not be necessary. So, we are able to divide our set of samples into two subsets: one for which the computation of $m \cdot m^2$ will induce a reduction and another for which it will not. If the value of $k_{\omega-2}$ is really 1, then we can expect the computation times for the messages from the first set to be slightly higher than the corresponding times for the second set.

On the other hand, if the actual value of $k_{\omega-2}$ is 0, then the operation $m \cdot m^2$ will not be performed. In this case, our "separation criterion" will be meaningless: there is indeed no reason why a m, inducing a reduction for the operation $m \cdot m^2$, would also induce a reduction for $m^2 \cdot m^2$, or for any other operation. Therefore, the separation into two subsets should look random, and we should not observe

[5] We can of course suppose that the first bit of the key is always 1.

any significant difference in the computation times.

Let us rewrite this a little more formally:

The algorithm $A(m,k)$ could be split into $L(m,k)$ and $R(m,k)$ where $L(m,k)$ is the computation due to the additional reduction at the multiplication phase for bit $k_{\omega-2}$ and $R(m,k)$ gathers all remaining computations. The total computation time is thus: $T(m) = T^L(m) + T^R(m)$, where $T^L(m)$, $T^R(m)$ are the times to compute $L(m,k)$ and $R(m,k)$ respectively.

The oracle O is:

$$O : m \rightarrow \begin{cases} 1 \text{ if } m \cdot m^2 \text{ is done with a reduction,} \\ 0 \text{ if } m \cdot m^2 \text{ is done without a reduction.} \end{cases}$$

As in section 7.3, define

$$\begin{aligned} &M_1 = \{m \in M : O(m) = 1\}, \\ &M_2 = \{m \in M : O(m) = 0\}, \\ &F_1 : M_1 \rightarrow \mathbb{R} : m \rightarrow F_1(m) = T(m), \\ &F_2 : M_2 \rightarrow \mathbb{R} : m \rightarrow F_2(m) = T(m). \end{aligned}$$

We have

$$\begin{cases} F_1 = T^R & \text{if } k_{\omega-2} = 0 \\ F_1 = T^R + T^L & \text{if } k_{\omega-2} = 1 \end{cases}$$

while,

$$F_2 = T^R$$

independently of the value of $k_{\omega-2}$.

Now, analyzing the mean as function ϕ, and testing:

$$\begin{aligned} H_0 &: \phi(F_1) \stackrel{?}{=} \phi(F_2) \\ H_1 &: \phi(F_1) \stackrel{?}{\neq} \phi(F_2) \end{aligned}$$

should reveal the value of $k_{\omega-2}$.

Once this value is known, we can simulate the computation up to the multiplication due to bit $k_{\omega-3}$, attack it in the same way as described above, and so on for the next bits.

7.4 Improvements and other targets

The timing attack described in previous section was further improved in [76, 78]. In its final form, it was able to recover a 512-bit key using between 5 000 and 10 000 timing measurements.

This attack assumes that the implementation does not use the CRT. In [77], Schindler presents an attack against an implementation using the CRT; this attack is very powerful (breaking a 1024-bit key with about 370 time measurements), but requires an adaptative adversary.

Block ciphers may be subject to timing attacks as well. In [38], Handschuh presents a timing attack against RC5, which requires 2^{20} measurements to succeed. In [78], we propose a timing attack against the AES (Rijndael), recovering a key with 4 000 measurements.

Timing attack can also be used in conjunction with other side-channel attacks. For example, Walter and Thompson [87] recently proposed very efficient attacks based on the analysis of time variations in RSA sub-operations. Their method requires to be able to observe partial timings, namely those of each individual multiplication of the exponentiation loop. This information cannot be obtained by sole time measurements, no matter how precise they are, but several other side channels (e.g. SPA) can be used for this purpose.

8 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 simple and differential power analysis, first introduced by Kocher et al. in [55].

As the clock ticks, the card's energy is also provided by the terminal, and can therefore easily be measured. 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.

Remark: the basic description of SPA and DPA is taken from the original paper on the subject [55].

8.1 Simple power analysis

Simple Power Analysis (SPA) is a technique that involves directly interpreting power consumption measurements collected during cryptographic operations.

SPA can yield information about a device's operation as well as key material.

A trace refers to a set of power consumption measurements taken across a cryptographic operation. For example, a 1 millisecond operation sampled at 5 MHz yields a trace containing 5000 points. Figure 2, for example, shows an SPA trace from a smart card performing a DES operation[6].

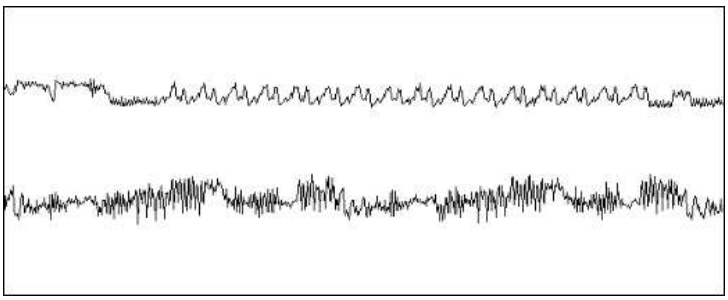

Figure 2: SPA monitoring from a single DES operation performed by a typical smart card. The upper trace shows the entire encryption operation, including the initial permutation, the 16 DES rounds, and the final permutation. The lower trace is a detailed view of the second and third rounds.

Because SPA can reveal the sequence of instructions executed, it can be used to break cryptographic implementations in which the execution path depends on the data being processed. For example:

DES key schedule: the DES key schedule computation involves rotating 28-bit key registers. A conditional branch is commonly used to check the bit shifted off the end so that "1" bits can be wrapped around. The resulting power consumption traces for a "1" bit and a "0" bit will contain different SPA features if the execution paths take different branches for each.

DES permutations: DES implementations perform a variety of bit permutations. Conditional branching in software or microcode can cause significant power consumption differences for "0" and "1" bits.

Comparisons: string or memory comparison operations typically perform a conditional branch when a mismatch is found. This conditional branching causes large SPA (and sometimes timing) characteristics.

[6]This figure is taken from [56].

Multipliers: modular multiplication circuits tend to leak a great deal of information about the data they process. The leakage functions depend on the multiplier design, but are often strongly correlated to operand values and Hamming weights.

Exponentiators: a simple modular exponentiation function scans across the exponent, performing a squaring operation in every iteration with an additional multiplication operation for each exponent bit that is equal to "1". The exponent can be compromised if squaring and multiplication operations have different power consumption characteristics, take different amounts of time, or are separated by different code. Modular exponentiation functions that operate on two or more exponent bits at a time may have more complex leakage functions.

8.2 Differential Power Analysis

In addition to large-scale power variations due to the instruction sequence, there are effects correlated to data values being manipulated. These variations tend to be smaller and are sometimes overshadowed by measurement errors and other noise. In such cases, it is still often possible to break the system using statistical functions tailored to the target algorithm.

Because of its widespread use, the Data Encryption Standard (DES) will be examined in detail. In each of the 16 rounds, the DES encryption algorithm performs eight S box lookup operations. The 8 S boxes each take as input six key bits exclusive-ORed with six bits of the R register and produce four output bits. The 32 S output bits are reordered and exclusive-ORed onto L. The halves L and R are then exchanged.

The DPA selection function $D(C, b, K_s)$ is defined as computing the value of bit $0 \leq b < 32$ of the DES intermediate L at the beginning of the 16th round for ciphertext C, where the 6 key bits entering the S box corresponding to bit b are represented by $0 \leq K_s < 2^6$. Note that if K_s is incorrect, evaluating $D(C, b, K_s)$ will yield the correct value for bit b with probability $P \simeq \frac{1}{2}$ for each ciphertext.

To implement the DPA attack, an attacker first observes m encryption operations and captures power traces $T_{1\ldots m}[1 \ldots k]$ containing k samples each. In addition, the attacker records the ciphertexts $C_{1\ldots m}$. No knowledge of the plaintext is required.

DPA analysis uses power consumption measurements to determine whether a key block guess K_s is correct. The attacker computes a k-sample differential

trace $\Delta_D[1 \ldots k]$ by finding the difference between the average of the traces for which a certain intermediate value V is one and the average of the traces for which V is zero. Thus $\Delta_D[j)$ is the average over $C_{1 \ldots m}$ of the effect due to the value represented by the selection function D on the power consumption at point j. In particular,

$$\begin{aligned} \Delta_D[j] &= \frac{\sum_{i=1}^{m} D(C_i,b,K_s)T_i[j]}{\sum_{i=1}^{m} D(C_i,b,K_s)} - \frac{\sum_{i=1}^{m}(1-D(C_i,b,K_s)T_i[j])}{\sum_{i=1}^{m}(1-D(C_i,b,K_s))} \\ &\simeq 2\left(\frac{\sum_{i=1}^{m} D(C_i,b,K_s)T_i[j]}{\sum_{i=1}^{m} D(C_i,b,K_s)} - \frac{\sum_{i=1}^{m} T_i[j]}{m}\right) \end{aligned}$$

If K_s is incorrect, the bit computed using D will differ from the actual target bit for about half of the ciphertexts C_i. The selection function is thus effectively uncorrelated to what was actually computed by the target device. If a random function is used to divide a set into two subsets, the difference in the averages of the subsets should approach zero as the subset sizes approach infinity. Thus, if K_s is incorrect,

$$\lim_{m \to \infty} \Delta_D[j] \simeq 0$$

because trace components uncorrelated to D will diminish with $\frac{1}{\sqrt{m}}$, causing the differential trace to become flat (the actual trace may not be completely flat, as D with K_s incorrect may have a weak correlation to D with the correct K_s).

If K_s is correct, however, the computed value for $D(C_i, b; K_s)$ will equal the actual value of target bit b with probability 1. The selection function is thus correlated to the value of the bit considered. Other data values, measurement errors, etc. that are not correlated to D approach zero. Because power consumption is correlated to data bit values, the plot of Δ_D will be flat with spikes in regions where D is correlated to the values being processed.

The correct value of K_s can thus be identified from the spikes in its differential trace. Four values of b correspond to each S box, providing confirmation of key block guesses. Finding all eight K_s yields the entire 48-bit round subkey. The remaining 8 key bits can be found easily using exhaustive search or by analyzing one additional round. Triple DES keys can be found by analyzing an outer DES operation first, using the resulting key to decrypt the ciphertexts, and attacking the next DES key. DPA can use known plaintext or known ciphertext and can find encryption or decryption keys.

8.3 Further results

Since the idea's first publishing, a large number of papers were devoted to power analysis attacks. This section will stick to some important ones[7].

Power attacks were quickly applied against RSA. As a simple example, one may simply observe that, if the power consumptions traces of a square and a multiply differ sufficiently to make them distinguishable, one can directly read the secret exponent from the square and multiply power trace (possibly averaged over several computations to reduce noise). Further results can be found in [60], and more recently in [26].

Akkar et al. later revisited the power analysis problem in [3]. They tried to determine the relative importance of power leakage due to instructions, manipulated data, ... and proposed a (pretty basic) model of the leakage. Based on some practical measurements, they also showed that Kocher's separation criterion (i.e. dividing the sample on the basis of some manipulated bit) was not optimal and proposed other criteria, that maximize the difference between power consumptions. This allows a factor 5 improvement in sample size needed, but requires a much better knowledge of the attacked implementation.

A more important improvement came with the appearing of high-order DPA [62] (although the idea of high order DPA was already mentioned in [55]). The idea is the following: taking as a basis some bit whose value will have to be computed during encryption, classical DPA is based on the idea that, at some point in the computation, the power consumption will be correlated to that bit's value. As a generalization, second-order DPA considers not one, but a pair of points in the consumption curve, and searches if the *joint* consumption is correlated to that bit value.

Let us illustrate this by an example: one of the countermeasures proposed against the DPA is called the duplication method [34]. This method consists in separating the data to be processed in two shares, which can easily be recombined (say, by XORing them, to fix ideas) to give back the original data. In this method, the encryption algorithm's implementation is modified in such a way that the two parts are processed separately and, at each step, the two parts' recombination would give back the current state of the encryption, although this

[7]For example, the AES effort induced several publications on the resistance of AES candidates. One must however not forget that side channel attacks aim *implementations* rather than algorithms themselves. Although it is true that some algorithms are easier to make immune than others, we do not believe this discussion was really relevant in the AES process. We will therefore not discuss this issue here.

re-combination is never performed, except at the very end. One can see that, with this method, the DPA cannot be applied, as the value of the bit b (with the notation of previous section) is never computed, and therefore the power consumption will be uncorrelated to that bit at any point in processing.

However, second-order DPA will be applicable to this case. Suppose that the value of bit b has been split in two nibbles, b_0 and b_1, the XOR of which yields b. Clearly, b_0's value cannot be predicted. On the other hand, it is clear that the computation of b_0 and b_1 are correlated. By considering the joint consumptions at the point where b_0 and b_1 are computed, it is possible to break the scheme.

This idea can of course be generalized to higher orders. However, Borst [16] notes that the complexity of the attack increases quickly with the order. On an intuitive point of view, this can for example be seen by observing that, in previous example, we do not know at which point the values b_0 and b_1 were computed, and we can not do much better than trying all possibilities.

Recently, a new variant of power analysis attack, named *template attack*, was proposed by Chari et al. [23]. According to the authors, this is the strongest form of side channel attack possible in an information theoretic sense. Whereas classical DPA reduces noise by averaging over a large number of sample (with the same key used), their approach focuses on precisely modelling noise, and using this to fully extract information present in a single sample. The need for a single sample allows to break implementations and countermeasures whose security is dependent on the assumption that an adversary cannot obtain more than one or a limited number of side channel samples. However, this attack requires that an adversary has access to an identical experimental device that he can program to his choosing; in this sense, it is therefore much less general.

9 Electromagnetic analysis attacks

Any movement of electric charges is accompanied by an electromagnetic field. The currents going through a processor can characterize it according to its spectral signature. *Electromagnetic attacks*, first introduced by Quisquater and Samyde [73], and further developed in [74, 31] exploit this side channel by placing coils in the neighbourhood of the chip and studying the measured electromagnetic field.

Timing attack, power analysis and electromagnetic analysis can be put in perspective as increasingly-dimensional side channels. Timing attack provides a single scalar (the running time) for each measurement. Power analysis provides

a vector showing, at each time unit, the corresponding power consumption. By allowing to choose the position of the (possibly multiple) coil(s) around the chip, electromagnetic analysis allows to build a 3-dimensional map of the magnetic field's evolution along time, thus providing 4-dimensional information. This allows for example to separate the contributions of various components of the chip, and therefore to study them separately.

The information measured can be analyzed in the same way as power consumption (simple and differential electromagnetic analysis – SEMA and DEMA), but may also provide much more information and are therefore very useful, even when power consumption is available[8]. Agrawal et al. [2] show 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, which have been somewhat neglected so far, 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 comes back to the field of tempest-like attacks [1]). Finally, they argue that EM emanations can even be used to break power analysis countermeasures, and illustrate this by sketching a practical example.

As another witness of the larger amount of information yielded by the electromagnetic field, and of its possible use in combination with power signal, Quisquater and Samyde recently showed [71] that it was possible to build a dictionary of instructions and their power/electromagnetic traces, and, using correlation techniques and neural networks, to recognize the instructions executed by a processor.

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.

[8] One can of course imagine contexts in which power consumption cannot be obtained, but where it is possible to measure the radiated field.

10 Countermeasures

10.1 Probing attacks protection

Being tightly related, hardware countermeasures against probing attacks have been presented together with the attacks themselves in section 5.

10.2 Fault attacks protection

10.2.1 Software countermeasures

The most obvious way that comes to mind in order to protect against fault attacks is to check the computation, for example by repeating it and comparing the results. Things are however not so obvious...

First of all, it must be noted that this policy is very costly, either in time (repeat computation) or in hardware (double hardware and perform both computations in parallel). Moreover, repeating the computation is not always satisfactory as, in the case of a permanent fault induction, it will yield identical, although wrong, results.

Another way to check for the presence of faults is, in the case of public-key cryptography, to verify the signature (or re-encrypt the message). This is usually less time-consuming, as the public exponent is usually chosen to be small. Unfortunately, this verification exponent is not always known by the device performing the signature (remember that a smart card has pretty limited resources).

Soon after the appearing of the fault attack against RSA, Shamir proposed (and patented!) a verification procedure [79] for RSA exponentiation. The idea is the following:

1. choose a small random number, r,
2. perform the two CRT branches modulo pr and qr. That is, compute $x_{rp} = m_{rp}^{d_{rp}} \bmod rp$, and $x_{rq} = m_{rq}^{d_{rq}} \bmod rq$, where the notations $m_{rp}, \ldots$ are directly adapted from alg. 1.
3. check if $x_{rp} = x_{rq} \bmod r$
4. compute and output $s = \text{chinese}(x_{rp} \bmod p, x_{rq} \bmod q)$

This procedure was improved by Joye, Paillier and Yen [43], who propose a more general and efficient algorithm. Further improvements were later proposed and patented by Gemplus.

However, Yen et al. [91] noted that Shamir's countermeasure was only effective if the fault is induced during the exponentiation phase of the CRT. If the fault occurs during the recombination phase (i.e. the function chinese() in alg. 1) then Shamir's countermeasure becomes useless. The mentioned paper propose new countermeasures taking this into account. This is also pointed out by Aumüller et al. [8], who experimented the attack against a card with all hardware countermeasures turned off. They also propose ways to avoid this weakness.

Another drawback of Shamir's countermeasure, as well as of all aforementioned computation checking methods, is that they involve a conditional instruction: two results are compared and the computation is considered correct only if this comparison is satisfying. The problem is that this conditional instruction may itself be subject to a fault attack[9]. For this reason, Yen et al. [91] propose a new countermeasure, called *fault infective* computation. As classical CRT, fault infective computation divides the computation in two branches, $\mathrm{mod}\, p$ and $\mathrm{mod}\, q$ (and recombines them with the CRT), but the division is made such that, if either branch is faulty, then, with very high probability, the recombined result will be incorrect *both* $\mathrm{mod} p$ and $\mathrm{mod} q$. The faulty result will therefore not allow to factorize n. We refer to [91] for more details. Note that [8] claims that this method can be completely broken by lattice reduction methods. Unfortunately, they only provide a "personal communication" reference for this claim, which makes it impossible to verify for the moment.

Concerning countermeasures based on computation correctness checking, Yen and Joye [49] have also noted that, even if this checking is not prevented from completing normally, it might still not be sufficient: as a matter of fact, they develop an attack that is able to recover a key by simply observing whether the device accepts to output the result or not. This attack's practicability is questionable, and we do not believe it constitutes a serious practical threat, but it is nonetheless a good illustration of the fact that great care has to be taken when implementing countermeasures: discarding incorrect results might not be sufficient.

Finally, we must point out that, as is noted by[9, 8], the fault induction attack against RSA requires a deterministic padding function. A simple countermeasure is therefore to use probabilistic schemes, as is the case of many current standards, such as the (PSS-based) PKCS#1 v2.1.

[9] In fact, several authors and manufacturers consider this type of fault induction much more practical than many of the ones (bit flip, ...) mentioned above.

10.2.2 Hardware countermeasures

Since the appearing of fault-based cryptanalysis, a number of papers proposing hardware countermeasures have been proposed. However, several among them have to be taken with great care: written by error-correction specialists rather than cryptographers, they sometimes miss the real issues of fault induction attacks. As an extreme example, we can cite [29], which basically adds some error-detection codes in the plaintext, such that error can be detected after decryption. Clearly, such a detection comes much too late to be of any use against fault attacks[10].

Karri et al. [51, 52] propose to add circuitry to perform, in parallel with the encryption, a reverting of the performed operations (with various possible levels of granularity) and to compare them with the input values to ensure that no error has occurred (this countermeasure is more oriented towards symmetriuc encryption). The remarks made in previous section about time or hardware costs apply here as well, and it is not clear that all possible attacks have been taken into account. The conclusion of [51] for example, states that it is assumed that the key RAM and the comparator cannot be subject to attacks.

In [90], Wang et al. propose a fault-resistant DES implementation, based on the addition of error-detecting codes to protect the integrity of registers. It is however not clear that this protection encompasses all possible fault inductions (e.g. faults affecting the processing of data, rather than their storing). In this sense, it is worth noting that the practical tests mentioned in the paper were performed by inducing only errors of the type considered in the model (namely, affecting the value stored in registers). Relevance of these tests is therefore disputable.

Hardware intrusion detection mechanisms can also be found in [59, 66].

More recently, an interesting idea is proposed by Moore et al. [65]. This paper, which proposes a smart card architecture resistant to side-channel attacks (the paper is not limited to fault induction attacks, but also takes into account timing attacks, power analysis, ...: we will come back on this in further sections) uses a self-timed circuit and dual rail-logic to thwart error induction. Roughly speaking, a self-timed circuit is characterized by the fact that its execution flow is not controlled by a central clock; instead, various components may operate at

[10] We owe it to the authors of this paper to mention that they do not present their result as a way to prevent fault induction attacks (a topic they do not seem aware of), but simply as a way to detect non malicious errors (e.g. transmission errors). This paper's usefulness is nonetheless void.

their own speed, warning their predecessors when they are ready to process data. Dual-rail encoding is often used to construct these circuits: two wires are used to encode three states: logic-0 (wire 1 to 0, wire 2 to 1), logic-1 (wire 1 to 1, wire 2 to 0) and clear (both wires to 0, meaning that the component is ready). Moore et al. propose to use the fourth state as an alarm. They argue[11] that a single-point fault induction (i.e. affecting only one wire) has a big chance to finally induce an alarm, and the card is wired in such a way that, once triggered, the alarm will propagate quickly throughout the device.

Although interesting, we would like to point out that this idea is rather new, and thus needs to receive scrutinity from the scientific community before it can be considered as an effective countermeasure.

It is also worth noting that the European project G3Card is dedicated to the design of a side-channel resistant smart card. Among other things, the conclusions of this project could therefore be of great interest regarding the relevance of the aforementioned countermeasures.

10.3 Timing attacks protection

Once the threat of timing attack has been identified, protecting applications against it is not a too difficult task[12].

Countermeasures can be of two types: hiding variations or blinding.

10.3.1 Hiding variations

The simplest way to hide variation is to make the computation strictly constant-time, for all possible secret exponents. This, however, would imply a very severe performance drawback, especially for asymmetric cryptosystems, since this constant time would of course be that of the slowest possible case (for RSA, for example, this would correspond to an exponent equal to $111\ldots1$). Such a countermeasure would therefore not be very practical.

Another possibility would be to modify the Montgomery algorithm so that an additional subtraction is always carried out, even if its result is simply discarded afterwards. This modification is easy to carry out, does not decrease performance

[11] See [65] for a more complete discussion.

[12] One must however not neglect the possibility for an attacker to obtain information about the running time by other means than pure time measurements; for example, even if no output has been produced yet, a variation in power consumption pattern may betray the fact that the device has completed its activity.

very much and clearly defeats the attack. One must however be very careful when implementing it and make sure to remove *all* time variation. For example, [27] notes that it turned out that the implementation they were attacking was using this countermeasure, but in too naive a way: additional subtraction was always carried out, which hid most time variation, but the difference between discarding and copying its result still induced a difference of some clock cycles. This countermeasure did not prevent the attack to be carried out, it simply made it a bit more difficult.

However, Dhem [28] proposed an improvement of these multiplications schemes, allowing several modular multiplications to be chained with only *one* extra reduction being performed after the last multiplication. This scheme seems to be especially interesting here, as it would suppress our attack's main target. Similar methods were proposed by Walter [88, 89] and by Hachez et al. [37], who study conditions under which a square and multiply algorithm can be carried out without the need for additional reductions between Montgomery multiplications.

One must however keep in mind that these countermeasures do not be guarantee that the system will be immune to *any* type of timing attack, but only against those which exploit the reduction of the multiplication algorithm.

Block ciphers are, generally speaking, easier to make constant time than asymmetric primitives. The two aforementioned attacks against RC5 and AES, for example, can easily be defeated, respectively by ensuring that the shift instruction's running time is not dependent on the number of positions to shift[13], and by ensuring that the timex operation will be implemented in a constant-time way (see [78] for more details).

An often proposed countermeasure is simply to add random delays to the algorithm, in order to hide time variation. We insist on the fact that this countermeasure is inefficient, as it is equivalent to adding white noise to a source. Such noise can easily be filtered out for an increase in sample size.

10.3.2 Hiding internal state

The second type of countermeasure consists in hiding internal state, so that the attacker cannot simulate internal computations any more. For example, [54] suggested the following blinding strategy: before computing the modular

[13] This can be done, either in hardware by implementing a constant-time shift operation, or in software by adding dummy instructions when necessary – with the already mentioned restrictions regarding the risk for these dummy instructions to be identifiable by other means than "pure" timing attacks.

exponentiation, choose a random pair[14] (v_i, v_f) such that $v_f^{-1} = v_i^e$; multiply the message by $v_i \pmod n$ and multiply back the output by $v_f \pmod n$ to obtain the searched result. As the attacker cannot simulate internal computations any more, she can hardly exploit her knowledge of the timing measurements.

Once again, this countermeasure is only guaranteed to defeat the type of attack we know, and we cannot rule out the possibility of a completely new timing attack, making blinding ineffective. In fact, some such attacks, combining timing and power analysis, have actually been proposed.

Blinding techniques are nonetheless an interesting direction in the search for countermeasures, and this not only against timing attack. As we showed in section 8.2, differential power analysis, as timing attack, requires the ability to predict internal states of the computation. Hiding these internal states can therefore be used as a countermeasure against DPA as well. We will come back to this issue in next sections.

10.4 Power analysis and electromagnetic analysis protection

As far as software countermeasures are concerned, electromagnetic attacks and power attacks (and, to a lesser extent, timing attacks) are, in many respects, very similar: the way the side channel leaks information differs, but the *type* of leaking information (i.e. timed information about the operations being processed or the data involved) is roughly the same. Software countermeasures do not try to reduce the signal amplitude, but rather to make the information it conveys useless by obscuring the internal parameters.

Designing hardware countermeasures against power analysis or electromagnetic analysis are different problems, but, these are nevertheless highly dependent: it is of no use to design a DPA countermeasure if this one facilitates EMA. This section will therefore discuss both questions together.

10.4.1 In a perfect world...

Ideally, countermeasures should be based on strong mathematical grounds. As Chari et al. [22] note, "*a scientific approach is to create a model for the physical characteristics of the device, and then design implementations provably secure in*

[14] [54] proposes a way to generate such pairs at a reasonable cost.

that model, i.e., they resist generic attacks with an a priori bound on the number of experiments" .

Several attempts have been made in this direction (e.g. [22, 3]), but, although they undeniably constitute useful starting points, none of them, in our opinion, proved really satisfactory, as the simplification assumptions they make often make them too distant from reality. For example, [22] explain that "*Each event's timing and power consumption depends on physical and environmental factors such as the electrical properties of the chip substrate, layout, temperature, voltage etc., as well as coupling effects between events of close proximity. As a first approximation, we ignore coupling effects and create a linear model, i.e., we assume that the power consumption function of the chip is simply the sum of the power consumption functions of all the events that take place*" . However, the unintentional emanations used in the attack of Agrawal et al. (see section 9) seem to be caused by coupling effects between components in close proximity. Although we are not exactly in the same context (electromagnetic analysis was not known at the time [22] was written), this shows that coupling effects cannot be neglected.

The design of *provably efficient* countermeasures is thus not possible yet, and we are left – as often in cryptography – with developing strongly motivated heuristics. In this sense, the aforementioned models are useful, as well as the experience gained by all previously discovered attacks.

10.4.2 Software countermeasures

To prevent SPA or SEMA, one has to make the execution flow as constant as possible, or, in other words, independent of the manipulated data. Several authors [16, 3] have shown that most of the power consumption (85%) is due to the instruction executed, compared to the data involved (10%). Preventing a program from taking different path depending on the data is therefore useful to make SPA or EMA more difficult. Similarly, it is probably good practice to use the same function to implement the square and multiply instructions in a S&M algorithm. Selecting instructions whose consumption profile is not too characteristic may also prove useful, but this has to be done with great care (we will come back to this in the section on hardware countermeasures).

Adding some randomness to prevent the attacker from reducing the noise by averaging over several runs is also a good idea. As a simple example, this can be done for RSA by adding a random multiple of $\phi(n)$ to the secret exponent before each exponentiation, so that the sequence of squares and multiplies will

be different for each run, even if this does not affect the final result. Such randomization techniques preventing noise reduction by averaging are also useful against differential attacks. However, in both cases, one must however keep in mind that:

- some attacks may be effective even with a single observation (this is for example the case of the aforementioned template attack[15] [23]. Similarly, Walter [84] showed that it could be possible to break RSA with one single exponentiation – we will come back on this below) and are therefore not prevented by randomization.
- the randomization itself might be subject to a power analysis. In the above example, the implementor must ensure that the addition phase of a multiple of $\phi(n)$ does not reveal the card's secret.

Several papers have been devoted to the case of RSA and square and multiply. One obvious way to hide the square and multiply sequence is to always perform a multiplication between squarings, discarding the result when it is not needed (taking care that the discarding operation cannot be distinguished). As mentioned before, this method is however time-consuming. Some authors suggested that m-ary RSA (in which consecutive bits are processed together), or sliding windows techniques could be used as countermeasures, since they make it more difficult to deduce the value of the secret exponent from the observed chain of squares and multiplications. However, Walter's Big Mac attack [84] shows that this is most probably insufficient. Walter further suggested other exponent recoding schemes [83, 86], with the goal to make it difficult to re-construct the secret exponent, even when knowing the chain of squarings and multiplications performed.

One must of course also take care that the exponentiation is not the only place through which secrets could leak: recently, Joye and Villegas [46] pointed out that, although most of the attention in countermeasures design has been turned to the exponentiation, several modular multiplication algorithms (and therefore the corresponding modular exponentiation algorithms) require a normalization step involving an integer division. They showed that this integer division can leak secret parameters, and proposed a SPA-resistant implementation.

[15] This does not mean that randomization techniques are useless against the template attack, but one must ensure that the random part is not under control of the attacker, even in test devices. We refer to the paper for more details.

Against DPA or DEMA, most software countermeasures work by preventing the adversary from predicting internal states of the computation. As we have seen, DPA works by partitioning the sample in two subsets based on the value of some (key-dependent) internal bit. Chari et al. [22] therefore propose a countermeasure "*based on well known* secret sharing *schemes where each bit of the original computation is divided probabilistically into shares such that any proper subset of shares is statistically independent of the bit being encoded and thus, yields no information about the bit*". This idea was first applied to RSA and DES by Goubin and Patarin [34]. For RSA, the idea is to replace the message m to be signed by a pair m_1, m_2 such that $m = m_1 m_2 \bmod n$, raise separately m_1 and m_2 to the secret power, and multiply them together to yield the desired result. The idea of DES secret sharing, although pretty simple, takes longer to describe, and we refer the reader to the original paper for more details. Similar ideas are discussed in [22].

As we showed in section 8.3, secret sharing methods can be defeated by applying higher-order DPA (or DEMA). However, Chari et al. (and Borst [16]) show that the complexity of the attack (in the sense of number of samples needed) grows exponentially with the number of shares.

One drawback of this method is its inefficiency, since the number of operations to perform is roughly doubled. As an alternative, some authors [25, 61, 33, 4] proposed *masking* methods. The idea is to combine the input with a random value, then to perform an operation with the key, and finally extract the random factor. For instance, adding subkey k to an intermediate result a can be implemented by the following operations

- choose r at random
- $z = a + r$
- $z = z + k$
- $z = z - r$

However, masking is difficult to apply to a full block cipher. For example, the masking explained in [61] uses "arithmetical" and "logical" masks and must, at some point in the computation, transform the first into the second. Akkar et al. [3] show that this transformation could be subject to their POPDA attack. Moreover, [32] shows that this countermeasure is not always useful: the authors present an attack against an addition implementation, based on the observation

of the Hamming weight of the sequence of carry that occur during the bitwise addition. Apart from this attack's efficiency[16], of more interest for us is the fact that this attack is not hindered by masking; in fact, the authors note that this could even make the attack easier.

10.4.3 Hardware countermeasures

Many authors (in fact, almost all those who present side channel attacks) discuss the basics of hardware countermeasures in side channel-related papers, but these discussions are often limited to very general principles.

On the other hand, smart card manufacturers are not very keen to giving details about the countermeasures they develop for their products (especially for hardware countermeasures). Patents (e.g. those taken by Paul Kocher) are an exception to this, and may be an useful information source.

Therefore, the information available in scientific literature is pretty limited on this issue.

The first idea that might be tempting against power attacks is to do some "balancing", i.e., try to negate the effects of one set of events by also performing another "complementary" set. Such approaches are however very difficult to perform perfectly, so, if the resolution is sufficiently high, or if the two complementary components are sufficiently distinct (so that an EMA will be able to distinguish between them), even slight differences might suffice to attack the implementation.

Another popular approach is to randomize the execution sequence i.e. keep operations the same, but permute the order (e.g. in DES, the S boxes are looked up in a random order). However, according to [22], "*unless this random sequencing is done extensively throughout the computation, which may be impossible since the specification forces a causal ordering, it can be undone and a canonical order recreated by signal processing. Even if the entire computation cannot be canonically reordered, it is sufficient to identify "corresponding" sample points in different runs so that a significant fraction are samples from the same power function P for the same cycle*".

A similar countermeasure consists in de-synchronizing the clock, for example by making instructions take a variable number of cycles or by having the cycles be of varying length. This will clearly complicate the attacker's task, but, provided

[16] Authors claim that IDEA can be broken with 2^{24} samples and between 2^{38} and 2^{56} computation effort. The attack is less efficient against other ciphers, such as Twofish or Mars.

the number of samples is sufficient, signal processing techniques might finally allow him to reconstruct the signal.

As mentioned in previous section, several authors suggest to use only instructions whose power consumption is not too characteristic. However, this selection must clearly be done with *all* side channels in mind. For example, [2] noted that some instructions leak much more information in some EM signals, compared to power signals (the opposite is also theoretically possible, but much less likely to occur in practice, since a designer who shields EM emanations is also likely to protect against power signal leakage).

Generally speaking, it is also important to make sure that a countermeasure against one specific attack does not make another one easier.

In [65], Moore et al. propose a smart card architecture aimed at resisting side-channel attacks in general. This architecture is based on self-timed circuits (we already briefly introduced this concept in section 10.2.2) and 1-of-n codes, basically meaning that several wires together carry a single information (1-of-2 code is equivalent to dual-rail logic, also discussed in section 10.2.2), which reduces data-dependent power consumption. Several other countermeasures, such as bus encryption, are also used. This paper is therefore a probable good starting point towards the design of tamper-resistant hardware, although one may regret the high-level of the description, with may technical details missing.

As we can see, none of the above countermeasures is perfect. The goal of the smart card designer must therefore not be to make the attacker's task impossible, but rather sufficiently difficult. In this sense, even naive countermeasures, such as the addition of white noise (which can clearly be filtered with sufficiently many samples) can prove useful.

10.5 Elliptic curve specific countermeasures

Elliptic curve cryptography allows some degrees of latitude which allow specific countermeasures (but also specific attack).

10.5.1 SPA protection

Several countermeasures against the SPA are proposed in the literature. They can mainly be put in four categories.

First of all, the obvious SPA countermeasure used for exponentiation schemes and consisting in always performing a squaring and a multiplication (discarding the result if necessary), can of course be adapted to elliptic curve cryptography,

in the form of an "always double and add" algorithm. However, this method presents of course the same efficiency drawback [81].

Countermeasures from the second category are based on the use of curves of specific form. Such countermeasures have been proposed for the Hessian form [44], for the Jacobi form [58][17] and for the Montgomery form [67, 68]. These countermeasures have the drawback that they cannot be applied to any type of elliptic curve. Oswald [69], for example, notes that the American National Institute of Standards and Technology (NIST) published a set of recommended curves which can be used as "named curves" in certificates and protocols. However, none of these curves has a Montgomery form, and therefore countermeasures using the Montgomery form cannot be applied to them.

A third category considers elliptic curves of the "classical" (i.e. Weierstrass) form, but use special point operation formulae. Among these we can cite the ones proposed by Izu and Takagi [41], Brier and Joye [17][18], Fischer et al. [30], and a result of Billet and Joye [12].

Finally, the last category is the countermeasures based on a method known as the Montgomery ladder [47, 81, 17].

Note that several of the above countermeasures can of course be combined.

10.5.2 DPA protection

Here too, the large number of proposed countermeasures can be divided in four main categories.

The first category, directly adapted from exponentiation schemes, consists in adding a multiple of the order of he curve before performing the scalar multiplication.

The second category consists in using different key-expansion methods. The use of sliding windows, for example, can be adapted from exponentiation schemes. Elliptic curves also offer other degrees of liberty in changing the key expansion. It is for example possible to use signed representations of the key: rather than a classical binary representation, this representation uses the three digits $1, 0, -1$. Countermeasures based on different key expansions can for example be found in [70, 40, 45].

As a third type of countermeasure, one can exploit the latitude in point representation: a point on an elliptic curves can be represented by various coordinates,

[17] Note that this countermeasure was recently partially broken by Walter [85].

[18] A paper of Izu and Takagi [42], to appear at PKC 2003, points out an attack against this countermeasure. However, we do not believe it to be really applicable in practice.

and changing the representation can be used in order to change the values manipulated by the program at each run, even if the underlying point (and therefore the result, from a cryptographic point of view) is exactly the same. Such countermeasures have been first proposed by Coron [24]. Among several other papers which were published on the subject, we point out the one by Joye and Tymen[19] [45]. Similarly, another countermeasure consists in randomizing the underlying curve rather than the point. Before performing the computation, the data are "transferred" on an isomorphic curve, where the actual computation takes place. Countermeasures based on this idea can be found in [45].

Finally, randomization can be obtained by exploiting isomorphisms on the underlying field. As a matter of fact, the (unique) field of characteristic p with p^n elements admits several representations. The idea, similar to the previous one, consists in choosing randomly a field $\mathbb{K}'$ isomorphic to $\mathbb{K}$ through isomorphism Φ, and computing kP as $Q = \Phi^{-1}(k\Phi(P))$. More details can be found in [45].

11 Conclusion

We believe to have shown that, at the moment, no perfect protection exists. By using appropriate countermeasures, it is possible to make the attacker's task harder (and therefore to limit the threat to more skilled, more resourceful, better trained adversaries) but not to make it impossible yet.

One must therefore start by defining the adversary the device must resist again, and the resources he disposes of, before choosing appropriate countermeasures against this adversary. Here as everywhere, security must be considered from an economical point of view too. Few people will spend US$ 1 000 000 to attack a device, the breaking of which can make them earn US$ 5 000.

This report tried to give an as extensive as possible overview of existing side-channel attacks, as well as possible countermeasure against them. Even if we tried to keep them in perspective, putting together all sorts of countermeasures is far from an easy task. Moreover, side-channel cryptanalysis is so implementation-specific that it is not possible to advise adequate countermeasure in a general framework. This topic needs therefore to be further studied in each practical case.

[19] The reader may have noted that this paper is cited in several categories. As a matter of fact, the paper proposes four different countermeasures, belonging to the four above categories.

Finally, we would like to point out that side channel attacks constitute a quickly evolving field, and countermeasures considered as very efficient three years ago may be made obsolete by today's research. It is therefore very important to keep informed on the evolution of this field.

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. [50].

[3] M.-L. Akkar, R. Bevan, P. Dischamp, and D. Moyart, *Power analysis, what is now possible...*, Advances in Cryptology - ASIACRYPT '00 (T. Okamoto, ed.), Lectures Notes in Computer Science (LNCS), vol. 1976, Springer-Verlag, 2000.

[4] M.-L. Akkar and C. Giraud, *An implementation of DES and AES, secure against some attacks*, in Çetin K. Koç et al. [18].

[5] Anderson, *Security engineering*, Wiley & Sons, New York, 2001.

[6] 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.

[7] ———, *Low cost attacks attacks on tamper resistant devices*, Proc. of 1997 Security Protocols Workshop, Lectures Notes in Computer Science (LNCS), vol. 1361, Springer, 1997, pp. 125–136.

[8] C. Aumüller, P. Bier, W. Fischer, P. Hofreiter, and J.P. Seifert, *Fault attacks on RSA with CRT: concrete results and practical countermeasures*, in Kaliski et al. [50].

[9] O. Benoît and M. Joye, *Protecting RSA against fault attacks*, Gemplus Labs smart news – June 2001. Available at `http://www.gemplus.com/smart/enews/st4/index.html`, 2001.

[10] I. Biehl, B. Meyer, and V. Muller, *Differential fault attacks on elliptic curve cryptosystems*, Advances in Cryptology - CRYPTO 2000 (M. Bellare, ed.),

Lectures Notes in Computer Science (LNCS), vol. 1880, Springer-Verlag, 2000.

[11] 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.

[12] O. Billet and M. Joye, *The jacobi model of an elliptic curve and side-channel analysis*, Cryptology ePrint Archive: Report 2002/125. Available at `http://eprint.iacr.org`.

[13] D. Bleichenbacher, *Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1*, Advances in Cryptology - CRYPTO '98 (H. Krawczyk, ed.), Lectures Notes in Computer Science (LNCS), vol. 1462, Springer-Verlag, 1998.

[14] 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`.

[15] 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.

[16] J. Borst, *Block ciphers: Design, analysis and side-channel analysis*, Ph.D. thesis, K.U.Leuven, 2001.

[17] E. Brier and M. Joye, *Weierstrass elliptic curves and 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.

[18] 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.

[19] 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.

[20] 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.

[21] S. Chari, C. Jutla, J. Rao, and P. Rohatgi, *A cautionary note regarding evaluation of aes candidates on smart cards*, Proc. second AES conference, 1999.

[22] ———, *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.

[23] S. Chari, J.R. Rao, and P. Rohatgi, *Template attacks*, in Kaliski et al. [50].

[24] J.-S. Coron, *Resistance against differential power analysis for elliptic curves cryptosystems*, in Çetin K. Koç and Paar [19].

[25] ———, *On boolean and arithmetic masking against differential power analysis*, in Çetin K. Koç and Paar [20].

[26] B. den Boer, K. Lemke, and G. Wicke, *A DPA attack against the modular reduction within a CRT implementation of RSA*, in Kaliski et al. [50].

[27] 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.

[28] J.F. Dhem, *Design of an efficient public-key cryptographic library for risc-based smart cards*, Ph.D. thesis, Université catholique de Louvain - UCL Crypto Group - Laboratoire de microélectronique (DICE), May 1998.

[29] S. Fernandez-Gomez, J.J. Rodriguez-Andina, and E. Mandado, *Concurrent error detection in block ciphers*, International Test Conference (ITC), IEEE, 2000.

[30] W. Fischer, C. Giraud, E.W. Knudsen, and J.P. Seifert, *Parallel scalar multiplication on general elliptic curves over $\mathbb{F}_p$ hedged against non-differential side-channel attacks*, Cryptology ePrint Archive: Report 2002/007. Available at `http://eprint.iacr.org`.

[31] 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.

[32] M. Gomulkiewicz and M. Kutylowski, *Hamming weight attacks on cryptographic hardware - breaking masking defenses*, Computer Security – ESORICS 2002 (D. Gollmann, G. Karjoth, and M. Waidner, eds.), Lectures Notes in Computer Science (LNCS), vol. 2502, Springer-Verlag, 2002.

[33] L. Goubin, *A sound method for switching between boolean and arithmetic masking*, in Çetin K. Koç et al. [18].

[34] L. Goubin and J. Patarin, *DES and differential power analysis: the duplication method*, in Çetin K. Koç and Paar [19].

[35] P. Gutmann, *Secure deletion of data from magnetic and solid-state memory*, Proc. of 6th USENIX Security Symposium, 1997, pp. 77–89.

[36] _____, *Data remanence in semiconductor devices*, Proc. of 7th USENIX Security Symposium, 1998.

[37] Gaël Hachez and Jean-Jacques Quisquater, *Montgomery exponentiation with no final subtraction: Improved results*, in Çetin K. Koç and Paar [20], pp. 293–301.

[38] Helena Handschuh, *Cryptanalyse et sécurité des algorithmes à clé secrète*, Ph.D. thesis, Ecole Normale Supérieure des Télécommunications, 1999.

[39] M.A. Hasan, *Power analysis attacks and algorithmic approaches to their countermeasures for Koblitz cryptosystems*, in Çetin K. Koç and Paar [20].

[40] K. Itoh, J. Yajima, M. Takenaka, and N. Torii, *DPA countermeasures by improving the window method*, in Kaliski et al. [50].

[41] 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.

[42] _____, *Exceptional procedure attack on elliptic curve cryptosystems*, Lectures Notes in Computer Science (LNCS), Springer, 2003, To appear.

[43] M. Joye, P. Paillier, and S.M. Yen, *Secure evaluation of modular functions*, International workshop on cryptology and network security (R.J. Hwang and C.K. Wu, eds.), 2001.

[44] M. Joye and J.-J. Quisquater, *Hessian elliptic curves and side-channel attacks*, in Çetin K. Koç et al. [18].

[45] M. Joye and C. Tymen, *Protections against differential analysis for elliptic curve cryptography*, in Çetin K. Koç et al. [18], pp. 377–390.

[46] M. Joye and K. Villegas, *A protected division algorithm*, in USENIX Association [82].

[47] M. Joye and S.M. Yen, *The Montgomery powering ladder*, in Kaliski et al. [50].

[48] 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.

[49] Marc Joye and Sung-Ming Yen, *Checking before output may not be enough against fault-based cryptanalysis*, IEEE Transactions on Computers **49** (2000), no. 9, 967–970.

[50] Burton S. Kaliski, Çetin K. Koç, and Christof Paar (eds.), *Cryptographic Hardware and Embedded Systems - CHES 2002*, Lectures Notes in Computer Science (LNCS), Springer-Verlag, August 2002.

[51] R. Karri, K. Wu, P. Mishra, and Y. Kim, *Concurrent error detection of fault-based side-channel cryptanalysis of 128-bit symmetric block ciphers*, DAC 2001, ACM 1-58113-297-2/01/0006, 2001.

[52] ———, *Fault-based side-channel cryptanalysis tolerant Rijndael symmetric block cipher architecture*, IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT'01), IEEE, 2001.

[53] V. Klima and T. Rosa, *Further results and considerations on side channel attacks on rsa*, in Kaliski et al. [50].

[54] 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.

[55] 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.

[56] P. Kocher, J. Jaffe, and B. Jun, *Introduction to differential power analysis and related attacks*, `http://www.cryptography.com/dpa/`, 1998.

[57] Olivier Kömmerling and Markus G. Kuhn, *Design principles for tamper-resistant smartcard processors*, Proc. of USENIX Workshop on Smartcard Technology (Smartcard '99), 1999.

[58] P.-Y. Liardet and N.P. Smart, *Preventing SPA/DPA in ECC systems using the Jacobi form*, in Çetin K. Koç et al. [18].

[59] 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.

[60] T. S. Messerges, E. A. Dabbish, and R. H. Sloan, *Investigations of power analysis attacks on smartcards*, Proc. USENIX Workshop on Smartcard Technology, 1999.

[61] Th. Messerges, *Securing AES finalists against power analysis attacks*, Fast Software Encryption: 7th International Workshop - FSE '00 (B. Schneier, ed.), Lectures Notes in Computer Science (LNCS), vol. 1978, Springer-Verlag, 2000.

[62] Th.S. Messerges, *Using second-order power analysis to attack DPA resistant software*, in Çetin K. Koç and Paar [20].

[63] B. Möller, *Securing elliptic curve point multiplication against side-channel attacks*, Information security – 4th international conference (ISC 2001) (Berlin) (G.I. Davida and Y. Frankel, eds.), Lectures Notes in Computer Science (LNCS), vol. 2200, Springer, 2001, p. 324 ff.

[64] P.L. Montgomery, *Modular multiplication without trial division*, Mathematics of Computation **44** (1985), no. 170, 519–521.

[65] S. Moore, R. Anderson, P. Cunningham, R. Mullins, and G. Taylor, *Improving smart card security using self-timed circuits*, Eighth IEEE International Symposium on Advanced Research in Asynchronous Circuits and Systems (IEEE, ed.), 2002.

[66] D. Naccache and D. M'Raihi, *Cryptographic smart cards*, IEEE micro, 1996.

[67] K. Okeya and K. Sakurai, *Power analysis breaks elliptic curve cryptosystems even secure against the timing attack*, Progress in Cryptology - INDOCRYPT 2000 (R. Bimal and E. Okamoto, eds.), Lectures Notes in Computer Science (LNCS), vol. 1977, Springer, 2000.

[68] _____, *Efficient elliptic curve cryptosystems from a scalar multiplication algorithm with recovery of the y-coordinate on a montgomery-form elliptic curve*, in Çetin K. Koç et al. [18].

[69] E. Oswald, *Enhancing simple power analysis attacks on elliptic curve cryptosystems*, in Kaliski et al. [50].

[70] E. Oswald and M. Aigner, *Randomized addition-subtraction chains as a countermeasure against power attacks*, in Çetin K. Koç et al. [18].

[71] J.-J. Quisquater and D. Samyde, *Automatic code recognition for smartcards using a Kohonen neural network*, in USENIX Association [82].

[72] _____, *Eddy current for magnetic analysis with active sensor*, Proc. of Esmart 2002, 2002.

[73] 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.

[74] _____, *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.

[75] W. Rankl and W. Effing, *Smart card handbook*, John Wiley & Sons, 1997.

[76] W. Schindler, *Optimized timing attacks against public key cryptosystems*, Statistics & Decisions (2000), to appear.

[77] ———, *A timing attack against RSA with the Chinese remainder theorem*, Proc. of Cryptographic Hardware and Embedded Systems (CHES 2000) (Ç. Koç and C. Paar, eds.), LNCS, vol. 1965, Springer, 2000, pp. 109–124.

[78] 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, December 2001, pp. 245–267.

[79] A. Shamir, *How to check modular exponentiation*, Presented at the rump session of EUROCRYPT '97, Konstanz, Germany.

[80] S. Skorobogatov and R. Anderson, *Optical fault induction attacks*, in Kaliski et al. [50].

[81] E. Trichina and A. Bellezza, *Implementation of elliptic curve cryptography with built-in counter measures against side channel attacks*, in Kaliski et al. [50].

[82] USENIX Association (ed.), *Fifth Working Conference on Smart Card Research and Advanced Applications (CARDIS '02)*, 2002.

[83] C.D. Walter, *Exponentiation using division chains*, IEEE Transactions on Computers (IEEE, ed.), vol. 47, 1998.

[84] ———, *Sliding windows succumbs to Big Mac attack,*, in Çetin K. Koç et al. [18].

[85] ———, *Breaking the Liardet-Smart randomized exponentiation algorithm*, in USENIX Association [82].

[86] ———, *MIST: An efficient, randomized exponentiation algorithm for resisting power analysis*, Topics in Cryptology - CT-RSA 2002, Lecture Notes in Computer Science, Springer, April 2002.

[87] C.D. Walter and S. Thompson, *Distinguishing exponent digits by observing modular subtractions*, Proc. of RSA conference 2001, 2001, to appear.

[88] Colin D. Walter, *Montgomery Exponentiation Needs no Final Subtractions*, Electronics Letters **35** (1999), no. 21, 1831–1832.

[89] ———, *Montgomery's Multiplication Technique: How to Make It Smaller and Faster*, in Çetin K. Koç and Paar [19], pp. 80–93.

[90] L.Y. Wang, C.S. Laih, H.G. Tsai, and N.M. Huang, *On the hardware design for DES cipher in tamper resistant devices against differential fault analysis*, IEEE international symposium on circuits and systems (IEEE, ed.), 2000.

[91] S.M. Yen, S. Kim, S. Lim, and S. Moon, *RSA speedup with residue number system immune against hardware fault cryptanalysis*, ICICS 2001 (K. Kim, ed.), Lectures Notes in Computer Science (LNCS), vol. 2288, Springer, 2001, pp. 397–413.