Side Channel Attacks on Implementations of Curve-Based Cryptographic Primitives

ROBERTO M. AVANZI*
Horst Görtz Institute for IT Security, Ruhr University Bochum, Germany
Roberto.Avanzi at ruhr-uni-bochum.de

January 23, 2005

Abstract

The present survey deals with the recent research in side channel analysis and related attacks on implementations of cryptographic primitives. The focus is on software contermeasures for primitives built around algebraic groups. Many countermeasures are described, together with their extent of applicability, and their weaknesses. Some suggestions are made, conclusion are drawn, some directions for future research are given. An extensive bibliography on recent developments concludes the survey.

Contents

*The work described in this paper has been supported by the Commission of the European Communities through the Fifth Framework Programme under Contract IST-2001-32613 (see `http://www.arehcc.com`).

1 Introduction

Any cryptographic primitive can be viewed as an abstract mathematical object: the primitive is in fact a function, or a set of functions parameterized by a key, for example from the set of plaintexts to the set of ciphertexts. On the other hand, the primitive will have to be implemented as a software or firmware program that will run on a given hardware, in a given environment, and will therefore present specific characteristics: just like the amount and type of noise produced by a car during its function can reveal something about speed and acceleration of the vehicle, there are types of information *leaked* from an implementation of the cryptographic primitive that can be used to elicit the secret information used by the primitive.

We can thus consider primitives both as mathematical objects and as concrete devices executing a program. These two points of view are very different, however both lead to approaches which can be useful to break a cryptosystem. In fact they can also be used together to create more powerful attacks. The first point of view is that of "classical" cryptanalysis; the second one is that of *side-channel cryptanalysis*. Even though side-channel cryptanalysis is specific to the implementation of the primitive to be broken, it is a very serious approach - in fact it has been used to break cryptosystems based on primitives which are considered mathematically secure.

The most popular target of side channel cryptanalysis, if not the very first, are smart cards. For the sake of simplicity, the discussion below will often be put in that context, although most of it applies to other cryptographic devices as well. The various side channels and related attacks known in the literature are briefly introduced. We then discuss countermeasures, and shall try to assess to which extent they are effective.

We must stress the fact that no perfect countermeasures exist. It would be very dangerous to believe that a given solution represents a perfect protection against all side channel attacks. Appropriate countermeasures can make the task of the attacker harder, to the point that, hopefully, he would exaust the resources he devoted to his goal. Therefore, when designing a device, we must define the adversary the device is supposed to resist - and only then an appropriate set of countermeasures can be chosen. In cryptography no amount of paranoia is too much: If our budget permits it, we can (read: should) use *more* countermeasures than what we deem *strictly* necessary, but *never*, ever, use less.

This field of research is a constantly moving target, hence embedded device designers should always keep themselves up to date.

The main types of attacks are: Invasive attacks, i.e. Probing and Fault induction attacks, and Non-Invasive attacks, i.e. Timing attacks and Leaked-Information attacks. The latter can use an analysis of power consumption or electromagnetic emissions, which are the most known side channels, but other side channels may be possible. All these attacks can be combined.

Here is a brief history of side channel and fault induction attacks.

- 1996: Timing Attacks [44].
- 1996: Fault analysis on RSA [79].
- 1997: Differential Fault Analysis (DFA) [11].
- 1998: Simple Power Analysis (SPA) [45, 46].
- 1998: Differential Power Analysis (DPA) (ibid.).
- 1999: Multiple Bit DPA [62, 63, 60] (but, see also [46]).
- 2000: Higher Order DPA [46, 59, 61].
- 2000: Adaptive Fault Analysis [96]. (See also [97].)
- 2002: "Discovery" of EM channel [2].
- 2003: Goubin-Type Analysis [27, 7].

The order in which the attacks and the countermeasures are presented in the sequel does not follow the chronological order. We have opted for an ordering in the exposition which, in our opinion, better shows the ideas underlying the attacks and introduces easier concepts first.

Our (partial) conclusions are: *almost all attacks that can be mounted against an elliptic curve cryptosystem can be mounted also against hyperelliptic curve cryptosystems – but the same holds also for a large number of the countermeasures, in fact rendering implementations of hyperelliptic curve cryptosystems as secure as implementation of elliptic curve cryptosystems*.

We also stress the fact that *there is a need for a sound theory of side channel cryptanalysis, including physical modelling of the devices, and a correct mathematical modelling of the attacks and of the proposed countermeasures.* A correct study of combinations of countermeasures is still missing.

We recall that the European project G3Card <http://www.g3card.org/> (Contract IST-1999-13515) is dedicated to the design of a tamper-proof smart card resistant to side-channel attacks, and that some side channel resistance results presented here have been supported also by the AREHCC project <http://www.arehcc.com/> (Contract IST-2001-31613).

The author acknowledges direct and indirect contributions by Marc Joye, François Koeune, Tanja Lange, and Prof. Dr. Jean-Jacques Quisquater.

2 Power and EM analysis

Side channel analysis was first introduced in the form of timing attacks in [44] and then simple and differential power analysis (SPA and DPA) [45, 46]. These attacks measure some leaked information of a cryptographic device (e.g. timing, power consumption,...) while it processes its inputs. From the monitored data, the attacker tries to deduce the inner-workings of the algorithm and thereby to retrieve some secret information.

A reasonable *model* for explaining the leakage is the following: Logic gates are made from transistors, and therefore draw current according to the number of switches active at any given moment. The amount of power drawn depends therefore in an indirect way on the operation being performed, the

operands being processed, data in the registers, other instructions in the pipeline... Characteristics of the processed operands that, in practice, have proven themselves correlated with the leaked traces are: the Hamming weight or individual bits, or bit sequences, of the processed operands (the paramount example being coordinates of points or divisors). Some attacks have exploited characteristics of the memory addresses used by the CPU of the device to fetch the information from memory.

One can put a resistor in series with the supply or the ground pin of the device and thus measure information which is correlated in real time to the internal state of the device. The measurements can be done at an astounding level of precision and resolution, even with relatively cheap hardware.

A 2002 IBM paper [2] caused a stir. It showed that one could do the same with the spectrum of electromagnetic (EM) emissions - and immediately after that it was revealed that the U.S. Gov. knew this much before (as a part of the research done in the classified TEMPEST project). But the history of publicly known results goes far before that. In fact, two symposia have been held in 1988 [81] and 1991 [82] in Rome about electromagnetic security for information protection. In 1996 some authors [4] expressed worries about the possibility of exploiting such emissions for breaking embedded devices, Quisquater and Samylde introduced simple and differential EM analysis at the Rump session of EUROCRYPT 2000 [77], and in 2001 Gandolfi et al. [26] presented some concrete results, showing how to disclose secret information used by cryptographic algorithms running on eight-bit embedded CMOS microcontrollers. EM analysis is very flexible: it can be detected at a long distance, however, the results are better if the emissions are recorded in close proximity to the device, possibly depackaged. EM emissions are localised, in other words an observer can not only measure the intensity of the activity in the smart card as a function of time, but also as a function of the space, at least in close proximity to the device.

In [80] we read "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". But, the EM emanations used in [2] seem to be indeed caused by the variations in electric current in the circuits as well as by coupling effects between components which are in close proximity inside the device. Hence, linear models need to be replaced. This shows the difficulty inherent to finding a sound physical model for the leakage.

We shall return later, in see Section 3, to attacks that exploit the total time of a cryptographic operation to infer information about the secret data being processed. We are concerned here with the attacks that are based on the analysis of power consumption or electromagnetic emission. If a single input is used, the process is referred to as a *Simple Side Channel Analysis* (SSCA), and if several different inputs are used together with statistical tools, it is called *Differential Side Channel Analysis* (DSCA).

Observing a single trace of leaked emissions one can reconstruct the sequence of elementary instructions performed by the processor, and thus infer the sequence of group operations. In an elliptic or hyperelliptic curve implementation with distinguishable group addition and doubling, and with a simple double-and-add scalar multiplication scheme, it is possible to reconstruct the secret scalar just from the observed sequence of distinct group operations. This is how SSCA works.

Suppose now that the group operations are indistinguishable one from the other by means of SSCA (this is always possible by inserting dummy operations). DSCA on a double-and-add scalar multiplication algorithm computing $n \cdot D$, with n fixed, secret, works as follows.

Let $n = (n_r, n_{r-1}, \ldots, n_0)$, and suppose $n_r, n_{r-1}, \ldots, n_{j+1}$ known. The attacker wants to find n_j. He proceeds as follows:

1. The attacker first makes a *guess*: $n_j = 0$ or 1.
2. He chooses several inputs $D_1, \ldots, D_t$ and computes $E_i = \left(\sum_{d=j}^{r} n_d 2^{d-j}\right) D_i$.
3. He picks a boolean selection function g to construct two index sets
$$\mathcal{S}_t = \left\{i \,:\, g(E_i) = \texttt{true}\right\} \quad \text{and} \quad \mathcal{S}_f = \left\{i \,:\, g(E_i) = \texttt{false}\right\} \ .$$
4. He puts $C_i = C_i(\tau) :=$ side-channel information obtained from the computation of $n \cdot D_i$. This is a function of the time τ. In the case where the EM emission is used, this function is indeed also a function of space, for example of a point on the surface of the card: $C_i = C_i(\tau; x, y)$.
5. Let $\langle C_i \rangle_{i \in \mathcal{S}}$ denote the average of the functions C_i for the $i \in \mathcal{S}$. If the guess of n_j was incorrect then
$$\langle C_i \rangle_{i \in \mathcal{S}_t} - \langle C_i \rangle_{i \in \mathcal{S}_f} \approx 0$$
i.e. the two sets are uncorrelated. On the other hand, if the guess of n_j was correct
$$\langle C_i \rangle_{i \in \mathcal{S}_t} - \langle C_i \rangle_{i \in \mathcal{S}_f}$$
as a function of time (and possibly space) will present spikes, i.e. deviations from zero.

2.1 Some words on hardware countermeasures

Since we are not evaluating hardware countermeasures here, we just mention some of the issues involved - which in practice further motivate the research on software countermeasures. The reader can skip this Subsection and return here after reading the rest of the present Section.

One of the best approach consists in designing hardware with constant power consumption [88] - this countermeasures is very expensive to implement. It provides protection also against attackers which can tamper and modify the device before performing the power signal measurements.

Shamir [85] proposes a different solution to the problem: to decorrelate the external power supplied to the card from the internal power consumption of the chip, by putting additional capacitors and/or batteries on the power supply path. By this, the power consumption trace is filtered, or smoothed out, and made essentially constant in time. This approach, far from perfect, in practice decreases the size of the power bias, thereby increasing the number of traces required for a successful DPA attack. For smart card applications it has been proposed also in [22] and in [78] for contactless devices. The underlying ideas are quite old: they have been used for almost a century in the development of power supplies for high-fidelity audio reproduction equipment. They are easy to implement, and are quite cost effective: only a few cents per card - we do not need high quality paper-oil silver foil capacitors as in for the so-called esoteric audiophile market! The drawback is that they are easily removed by an attacker performing active probing and microsurgery on the device.

The same considerations apply also to EM emissions. The design of hardware with instrinsically constant EM emissions is extremely expensive. EM shielding is effective, but only against passive attackers.

One obvious software countermeasure against differential side channel attacks (Subsection 2.4) is the usage of *random process interrupts* (RPIs). Instead of executing all the operations sequentially,

the CPU interleaves the code's execution with that of dummy instructions so that corresponding operation cycles do not match because of time shifts. This has the effect of smearing the peaks across the differential trace due to a desyncronisation effect, known in digital signal processing under the name of *incoherent averaging* [56, pp. 327–330]. The time shifts can be considered as added noise. RPIs do not render differential side channed analysis theoretically infeasible but increase the number of samples considerably. This goes far beyond dummy operations (Subsubsection 2.3.1) which are a software countermeasure, because it is implemented at the hardware level, and makes the operation flow intentionally irregular - instead of regular. These two countermeasures can be used together. This should be combined with suitable randomisation of the scalar if that is fixed. Microprocessors which are capable of randomized multithreading can be effectively used to implement this type of software countermeasures, because sometimes *dummy* traces will be added to effective ones, other times only effective ones will be added, and the position of the activity in the processing unit varies in an unpredictable way, making EM analysis more difficult. This is an example where special hardware can be used to implement more powerful software countermeasures. See [19] for attacks on devices featuring random process interrupts and noisy power consumption. Clocking devices using a randomized clock signal produces a similar protection [47]. A processor implemented all these countermeasures (RPIs on parallel pipelines and random time shiftings) is described in [57]. RPIs and randomized clocking make adaptive fault analysis (See Subsection 4.2) very difficult if not, in practice, impossible.

Conventional countermeasures based on the addition of random noise in the power profile exploit redundant hardware which, in turn, increases the power consumption. Benini et al. [9], building on their expertise in designing low consumption hardware, devised a hardware technique called *randomized power masking*, which is based on a combination of power-management techniques and randomized clock gating. By this they introduce in the hardware a significant amount of non-deterministic noise in the power profile, at no cost in the circuit average power consumption. Their approach increases the area used by the device.

The best hardware countermeasures may require increased power consumption or increased silicon area, but smart cards must be built under very strict specifications, so that they are not allowed to draw more current than a specified value, and they must fit in a predefined area. This is unfortunate, since these specifications had been determined for early, simple, and very insecure, hardware...

2.2 Software countermeasures

Software countermeasures do not try to avoid leakage (for example by making the power consumption constant). Instead, they attempt to make the information it conveys useless. Countermeasures at the software level seem to be more desirable than hardware countermeasures, from a commercial standpoint at least, since they can be implemented on existing architectures If effective, they also allow smart card developers to design and build *new* cheaper hardware - or to reduce investments on hardware countermeasures to concentrate more on increasing raw performance. Hardware countermeasures may be necessary anyway depending on the required security level.

As far as software countermeasures are concerned, electromagnetic attacks and power attacks are, in many respects, very similar: the way the side channel leaks information differs, but the nature of the leaked information is roughly the same. From this point of view, timing attacks bear some similarity to differential power analysis.

The rest of thie section is devoted to the software countermeasures, and to more refined attacks together with the corresponding protections.

2.3 Simple side channel analysis

To harden a cryptographic primitive against simple side channel analysis one can follow one of the following three approaches: dummy arithmetic instructions, indistinguishable or unified addition and doubling formulae, and the adoption of a scalar multiplication algorithm with fixed sequence of group operations.

2.3.1 Dummy arithmetic instructions

Inserting *dummy arithmetic instructions* in the group operations [21] is the "universal" way of making the operation flow homogeneous. This is most attractive when the two basic operations are already quite similar (elliptic curves in affine coordinates are a paramount example, but also hyperelliptic curves of genus 2 in most cases). It is usually a quite expensive countermeasure in all other cases.

Recently Chevallier-Mames et al. [16] have introduced a new approach: They *split* each of addition and doubling in more elementary basic blocks, and then make those homogeneous. This method has the advantage that no operation is unnecessarily made a lot longer by introducing several operations. The potential drawback is that it requires a very careful implementation in order to avoid "breaks" between basic blocks sequences of different lengths, which would still enable an attacker to distinguish the different group operations. This approach seems very promising, in particular on implementation of cryptographic primitives where doubling and addition have very different lengths [76, 75]. Note that, in this case, the measurement of the total time of the cryptographic operation (Section 3) can still allow the attacker to obtain the total Hamming weight of the scalar. The performance penalty is rather small.

Dummy instructions alone however are not a sound countermeasure against all types of side channel analysis. In fact, fault attacks can be used (see Section 4) to reveal the dummy operations and thus to distinguish the operations. A scalar randomization procedure, or the use of additional hardware which reveals the induction of faults, are therefore required (see Subsubsection 2.4).

2.3.2 Indistinguishable or unified addition and doubling formulae

This approach has been pursued until now only for elliptic curves. It consists in developing formulae for addition and doubling which are indistinguishable or unified [14, 15]: this may go as far as considering alternative, equivalent forms, of the elliptic curve, on which the addition and doubling formulae are equal.

One of the strongest points of this type of countermeasure is that there are *no* dummy operations involved, hence adaptive fault analysis techniques cannot be used (see Section 4).

The only drawback of some these approaches is that they are *very* specific to the type of group selected. Also, if the curve must be defined by an equation which is different from the Weierstraß form, it may be the case that not all curves which are defined in standards can profit from such techniques [12, 37, 53].

2.3.3 Scalar multiplication with fixed sequence of group operations

To prevent SSCA one can of course use scalar multiplication algorithms that have a fixed sequence of group operations, independent from the scalar [66, 69, 55, 70, 64, 34].

The usual trick for making the double-and-add algorithm homogeneous consists in performing a dummy group addition when the multiplier bit is zero. Therefore, the scalar multiplications appears

to an attacker observing the side channel information as an alternating sequence of doublings and additions. This performance penalty is of course huge. The same idea can be applied to scalar multiplication using a NAF-representation [30] – see also [17, Page 105]. In the latter case the sequence of operations looks as a repeating block of type "doubling, doubling, addition and doubling". This countermeasure is furthermore subject to the same type of fault analysis that can break dummy arithmetic operations - hence it requires the same kind of additional precautions, such as scalar randomization of the type described in § 2.4.1.b.

The so-called Montgomery form of an elliptic curve [66, 69, 55] has unified operations for computing, given two points P and Q for which $P - Q$ is known, both $P + Q$ and $2P$. Additionally, this technique does not use one of the coordinates of the points, hence do not need to recompute it a every step, making the computations very efficient. It has been adapted to curves in Weierstraß form [1, 24], too. See also [40] and [14, 15]. It would be interesting to have a similar form for hyperelliptic curves.

2.4 Differential side channel analysis

A scalar multiplication algorithm which is protected against simple side-channel analysis may still be vulnerable to differential analysis. From a practical point of view, however, very few public-key group-based cryptosystems are susceptible to such attacks: In fact in most cases the base group element is imposed by the system and the multiplier is an ephemeral parameter, varying at each execution. In some cases, however, the multiplier is the secret key and is fixed, and the base element is the input to the system. In such a scenario, DSCA becomes a threat.

DSCA is based on the knowledge of the internal representation of the operands, hence the countermeasures will work by "scrambling" all the bits of the computation in a (hopefully) unpredictable way. Usually, the inputs of the point multiplication algorithm, namely the base group element and the multiplier will be randomized: see [39] for elliptic curves and [7] for hyperelliptic curves. We now review the techniques and their advantages and disadvantages.

2.4.1 Scalar randomization

2.4.1.a Varying the representation

The insertion of random decisions during the execution of the point multiplication algorithm also helps in preventing side-channel analysis. Usually these decisions amount to randomly choosing among several different redundant weighted binary representations of the same integer [28, 74]. Such methods must be used with care, and indeed their soundness has been questioned [93], sometimes under the assumption that no SSCA-countermeasures are implemented [72, 73].

Liardet and Smart [53] describe three general-purpose randomised signed windows methods for performing SCA-resistant scalar multiplication. Under the hypothesis that additions are distinguishable from doublings, Walter [90] shows that repeated use of the same secret key with the algorithm of Liardet and Smart is insecure. Another randomized m-ary method is given in [33].

It seems that randomised redundant representations of integers are on the whole less safe than the standard, minimal weight, m-ary recodings. In fact, several different redundant representations of the same scalar are likely to give information on individual bits or bit sequences within the scalar.

Also the results of [89] apply to this context (for a security analysis, see [92]): even though the exponentiation algorithm there is described for RSA-like systems, the adaptation to an arbitrary group based cryptographic system is straightforward. The basic idea is to randomly choose between several different representations on a mixed base number system, at the same time selecting small addition

chains for the different parts of the computation which are composed from similar sub-blocks of group operations. Ideally such a technique would provide both SSCA and DSCA resistance. However, especially in presence of long keys, also this method alone can leak some information on the exponent [91].

Ciet, Quisquater and Sica [18] show how to efficiently randomize the decomposition of the scalar in the GLV scalar multiplication algorithm [25, 86]. This method applies also to hyperelliptic curves.

2.4.1.b Varying the scalar

For RSA, the standard method [44] is to add a random multiple of $\phi(N)$, where N is the RSA modulus, to the secret exponent before each exponentiation, so that the sequence of squares and multiplies will be different for each run. Of course, this does not affect the final result. For group based cryptography, the same idea amounts to replacing the secret scalar n with $n + k\ell$ in nD for a random integer k, where ℓ is the order of the group used in the system. In [70] it has been shown that this randomization may leave a bias in the least significant bits of the scalar. Möller [64] combines it (only in the `ecc` case) with an idea of Clavier and Joye, and suggests the computation of $nD = (n + k_1 + k_2\ell)D - k_1D$, where k_1 and k_2 are two suitably sized random numbers. In the context of group-based cryptography this method is called Coron's first countermeasure [21]. It induces a performance penalty depending linearly on the bit lengths random quantities k, k_1, k_2.

These countermeasures can be combined with any of those described in § 2.4.1.a to provide stronger defence.

2.4.2 Randomization of the base group element

There are essentially four types of randomization (blinding) of the base group element. The first one exploits the knowledge of the fixed scalar (before randomization), whereas the second and third family exploit redundancy of the representation of the group elements or of field elements. The fourth approach randomizes the whole computation, curve and points or divisors.

2.4.2.a Randomization exploiting knowledge of the fixed scalar

This is called Coron's second countermeasure [21]. We describe it in full generality. The countermeasure consists in replacing the computation of nD with that of $n(D + R) - S$, where R is a secret element for which $S = nR$ is known. It is desirable to have a new pair (R, S) at each run of the device. To do this, the following approach can be taken. A set of secret pairs (R_i, S_i) with $S_i = nR_i$ can be stored in the smart card at *initialization time*, and at each run both elements of a randomly chosen pair are multiplied by the same small signed scalar and added to the respective elements of another pair. The result is then used to randomize the scalar multiplication.

This is particularly useful if applied in conjunction with other countermeasures to thwart Goubin's attack (see Subsection 2.5 below).

2.4.2.b Randomization exploiting redundancy of representations

For elliptic curves and genus 2 hyperelliptic curves there exist *redundant* coordinate systems, which allow a point to be represented in many different ways. Such coordinate systems have been developed to allow computations in the group without having to perform inversions. Inversions are extremely expensive in embedded devices, so it makes sense to trade them for some multiplications

and a slightly larger memory usage. See [20] for a review of the coordinate systems available for elliptic curves and [48, 49, 50, 51] for the case of hyperelliptic curves. In fact, as [29] and [6] show, the usage of these coordinate systems is advantageous even in software implementations.

These representations, however, have also another advantage. In elliptic projective coordinates two triples (X, Y, Z) and (sX, sY, sZ) represent the same point if $s \neq 0$. In elliptic Jacobian coordinates two triples (X, Y, Z) and (s^2X, s^3Y, sZ) represent the same point if $s \neq 0$. Similar results hold for the other coordinate systems described in [20]. Therefore, to avoid differential side channel analysis on elliptic curves, the computations can be performed in projective (resp. Jacobian) coordinates, and the base point (X, Y, Z) can be replaced by (sX, sY, sZ) (resp. (s^2X, s^3Y, sZ)) for a random, non-zero field element s before starting the scalar multiplication. This approach is first described in [39].

Avanzi [7] has carried this idea over to the genus 2 setting. Let the hyperelliptic curve $\mathcal{C}$ of genus g over the finite field $\mathbb{F}_q$ be defined by the *Weierstraß equation*

$$\mathcal{C} \ : \ y^2 + h(x)y = f(x) \ ,$$

where f is monic of degree $2g + 1$ and h has degree at most g. Let ∞ be the point at infinity on the curve. The elements of the divisor class group in Mumford's representation [68] are given by polynomial pairs, and there are algorithms and explicit formulae for computing with those. Let $D = \sum_{i=1}^{m} P_i - m\infty$ be a reduced divisor with $m \leqslant g$, the P_i being points on the curve. The ideal class associated to D is represented by a unique pair of polynomials $U(x), V(x) \in \mathbb{F}_q[x]$ with $g \geqslant \deg U > \deg V$, U monic and such that: $U(x) = \prod_{i=1}^{m}(x - x_{P_i})$ (i.e. the roots of $U(x)$ are the x-coordinates of the points belonging to the divisor); $V(x_{P_i}) = y_{P_i}$ for all $1 \leqslant i \leqslant m$ (i.e. the polynomial $V(x)$ interpolates those points); and $U(x)$ divides $V(x)^2 - h(x)V(x) - f(x)$. We say that the pair $[U(x), V(x)]$ represents the divisor class of D.

In genus 2 projective coordinates a divisor class D with associated reduced polynomial pair $[U(x), V(x)]$ is represented as a quintuple $[U_1, U_0, V_1, V_0, Z]$ where

$$U(x) = x^2 + \frac{U_1}{Z}\,x + \frac{U_0}{Z} \quad \text{and} \quad V(x) = \frac{V_1}{Z}\,x + \frac{V_0}{Z} \ .$$

The randomization in this case consists in picking a random $s \in \mathbb{F}_q^*$ and by performing the following replacement

$$[U_1, U_0, V_1, V_0, Z] \mapsto [sU_1, sU_0, sV_1, sV_0, sZ] \ .$$

In Lange's genus 2 *new* coordinates a divisor class is represented as a sextuple $[U_1, U_0, V_1, V_0, Z_1, Z_2]$ where

$$U(x) = x^2 + \frac{U_1}{Z_1^2}\,x + \frac{U_0}{Z_1^2} \quad \text{and} \quad V(x) = \frac{V_1}{Z_1^3 Z_2}\,x + \frac{V_0}{Z_1^3 Z_2} \ .$$

To blind the base point or an intermediate computation, two elements s_1, s_2 are picked in $\mathbb{F}_q^\times$ at random and the following substitution is performed:

$$[U_1, U_0, V_1, V_0, Z_1, Z_2] \mapsto [s_1^2 U_1, s_1^2 U_0, s_1^3 s_2 V_1, s_1^3 s_2 V_0, s_1 Z_1, s_2 Z_2] \ .$$

If (some or all of) the additional coordinates z_1, z_2, z_3 and z_4 are used – which satisfy $z_1 = Z_1^2$, $z_2 = Z_2^2$, $z_3 = Z_1 Z_2$ and $z_4 = z_1 z_2$ – then they must also be updated: the fastest way is to recompute them from Z_1 and Z_2 by two squarings and two multiplications.

Ciet and Joye also suggested [17] a clever way to use point (or, we add, divisor) randomization while getting high performance. They suggest to randomise the intermediate value but not the base point in the scalar multiplication. For example, the computation of $n \cdot D$ where $n = (n_r, n_{r-1}, \ldots, n_0)_2$ with $n_r = 1$ may be performed by a double-and-add method as follows

```
Q = randomise(D);
for i = r-1 to 0 do
    Q = 2 Q;
    if (n_i == 1) then Q = Q + D; else dummy addition.
end for;
output Q;
```

In the case of elliptic curves, the base point may be given in affine coordinates, but `Q` would be in projective or jacobian coordinates, and `Q` would also be randomised in the chosen coordinate system. It can be adapted to several different scalar multiplication algorithms. Ciet and Joye call this technique $2P^*$.

If addition and doubling are indistinguishable, then this method is a good starting point for hardening the cryptosystem against DSCA. However, some work still has to be done against operand reusage detection using higher-order EM emission DSCA (see Subsection 2.6).

2.4.2.c Randomization using isomorphic fields

This randomization has been described by Joye and Tymen in [39]. It has been described for elliptic curves over binary fields, but it actually works for any Abelian group over any extension field. It consists in representing the field used for the arithmetic, $\mathbb{F}_{p^m}$, with $m > 1$, as the quotient ring $\mathbb{F}_p[x]/(f(x))$ for randomly updated irreducible polynomials $f(x)$ over $\mathbb{F}_p$ of degree m. This randomization effectively scrambles almost all bits of the computations, except for the least significant one. The main drawback is that the field arithmetic can become excruciatingly slow. In fact, usually the irreducible polynomial is taken to be a trinomial, a pentanomial or a sedimentary polynomial in even characteristic, or a binomial in odd characteristic [5], but after randomization it is usually a dense polynomial, and this makes polynomial multiplication slower by a factor of 3 or even more.

2.4.2.d Randomization using isomorphic groups

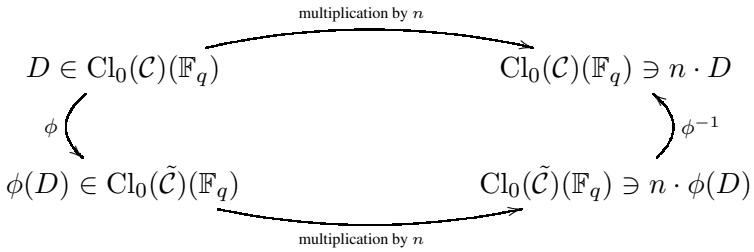
Also this approach has been introduced in [39] for elliptic curves, and generalized to the hyperelliptic curve setting in [7]. We describe it in the latter version, which includes the first in the particular case of genus 1 curves.

Let $\mathcal{C}$ and $\mathcal{C}'$ be two hyperelliptic curves of genus $g \geqslant 1$ over a finite field $\mathbb{F}_q$. Suppose that $\phi : \mathcal{C} \to \tilde{\mathcal{C}}$ is an $\mathbb{F}_q$-isomorphism which is easily extended to an $\mathbb{F}_q$-isomorphism of the divisor class groups $\phi : \mathrm{Cl}_0(\mathcal{C}) \to \mathrm{Cl}_0\left(\tilde{\mathcal{C}}\right)$. Let us further assume that ϕ, together with its inverse ϕ^{-1}, is computable in a reasonable amount of time, *i.e.* small with respect to the time of a scalar multiplication. We do not require *a priori* the computation time of ϕ to be negligible with respect to a single group operation. Then instead of computing $Q = nD$ in $\mathrm{Cl}_0(\mathcal{C})(\mathbb{F}_q)$, where n is an integer and $D \in \mathrm{Cl}_0(\mathcal{C})(\mathbb{F}_q)$, we perform:

$$Q = \phi^{-1}\big(n\,\phi(D)\big)$$

so that the bulk of the computation is done in $\mathrm{Cl}_0\left(\tilde{\mathcal{C}}\right)(\mathbb{F}_q)$, or, since a picture is worth a thousand

words, we note that the following diagram commutes:



and we follow it along the longer path. The countermeasure is effective if the representations of the images under ϕ of the curve coefficients and of the elements of $\mathrm{Cl}_0(\mathcal{C})(\mathbb{F}_q)$ are unpredictably different from those of their sources. This can be achieved by using randomly chosen isomorphisms ϕ, which boils down to multiplying all the quantities involved in a computation with "random" numbers.

Let $\mathcal{C}, \tilde{\mathcal{C}}$ be two hyperelliptic curves of genus g defined by *Weierstrass equations*

$$\mathcal{C} \ : \ y^2 + h(x)y - f(x) = 0 \tag{1}$$

$$\tilde{\mathcal{C}} \ : \ y^2 + \tilde{h}(x)y - \tilde{f}(x) = 0 \tag{2}$$

over $\mathbb{F}_q$, where $f, \tilde{f}$ are monic polynomials of degree $2g+1$ in x and $h(x), \tilde{h}(x)$ are polynomials in x of degree at most g. All $\mathbb{F}_q$-isomorphisms of curves $\phi : \mathcal{C} \to \tilde{\mathcal{C}}$ are of the type

$$\phi \ : \ (x, y) \mapsto \left(s^{-2}x + b, s^{-(2g+1)}y + A(x)\right) \tag{3}$$

for some $s \in \mathbb{F}_q^\times$, $b \in \mathbb{F}_q$ and a polynomial $A(x) \in \mathbb{F}_q[x]$ of degree at most g. Upon substituting $s^{-2}x + b$ and $s^{-(2g+1)}y + A(x)$ in place of x and y in equation (2) and comparing with (1) we obtain

$$\begin{cases} h(x) = s^{2g+1}\Big(\tilde{h}\big(s^{-2}x + b\big) + 2\,A(x)\Big) \\ f(x) = s^{2(2g+1)}\Big(\tilde{f}\big(s^{-2}x + b\big) - A(x)^2 - \tilde{h}\big(s^{-2}x + b\big)A(x)\Big) \end{cases} \tag{4}$$

whose inversion is

$$\begin{cases} \tilde{h}(x) = s^{-(2g+1)}h(\hat{x}) - 2A(\hat{x}) \\ \tilde{f}(x) = s^{-2(2g+1)}f(\hat{x}) + s^{-(2g+1)}h(\hat{x})A(\hat{x}) - A(\hat{x})^2 \\ \qquad\qquad \text{where} \quad \hat{x} = s^2(x - b) \ . \end{cases} \tag{5}$$

Avanzi shows that

1. For hyperelliptic curves in odd characteristic we can restrict the isomorphisms to the type

$$\phi \ : \ (x, y) \mapsto \left(s^{-2}x, s^{-(2g+1)}y\right) \ , \tag{6}$$

with $h(x) = \tilde{h}(x) = 0$. This gives a fast countermeasure which effectively multiplies all quantities involved in the computations of the group operations by powers of the randomly chosen non-zero parameter s.

2. For genus 2 hyperelliptic curves in characteristic 2, curve randomization is not adequate if one wants to keep some restrictions on the coefficients of $h(x)$ and $\tilde{h}(x)$ which are reasonable for performance reasons. But in this case group element randomization can be used.

3. In the genus 3 case curves of equation $y^2 + cy = f(x)$ can be randomized obtaining good performance and security - thus effectively obviating the lack of divisor randomization in this setting.

2.5 Goubin-type analysis

Recently L. Goubin [27] has pointed out a potential weakness of some `ecc` randomization procedures, including Coron's third and Joye-Tymen's, when implemented on systems where the secret scalar is fixed and the base of the scalar multiplication (the message) can be chosen.

His attack is based on the randomization of 0 by multiplication by a constant or by field isomorphism being still 0. It requires that the scalar multiplication algorithm has a fixed sequence of group operations for a given scalar – even after removing any dummy operations.

In full generality, we have the following context:

Let G be the group where the computations take place, and let H be a small subset of the group G s.t.:

- The elements of H possess properties which make their processing detectable by side-channel analysis – for example, zeros in the internal representation – and
- are invariant under a given randomisation procedure $\mathcal{R}$.

The set H is called the set of *special* group elements. It is used in an attack as follows:

Suppose that the most significant bits $n_r, n_{r-1}, \ldots, n_{j+1}$ of the secret scalar n are known and that we want to discover the next bit n_j.

1. The attacker first makes a *guess*: $n_j = 0$ or 1.
2. Set up a *chosen message attack* to obtain – if n_j has been guessed correctly – an intermediate result in H during the scalar multiplication. For example, let $t = (n_r, n_{r-1}, \ldots, n_{j+1}, n_j)$. The attacker chooses first $E_i \in H$, for $i = 1, 2, ..., w$, where w is the number of traces he wants to analyze. Then he finds elements $D_i \in G$ with $t \cdot D_i = E_i$ (i.e. $(t^{-1} \bmod \#G) \cdot E_i$), and uses each D_i as input. A further requirement is that for the other choice(s) of n_j the corresponding multiples of D_i should not be elements of H.
3. Then, statistical correlation of the side-channel traces corresponding to the computations of $n \cdot E_i$ may reveal if the guess was correct even if the randomisation procedure $\mathcal{R}$ has been used.

As already mentioned, such sets H exist:

- The set of points with a zero coordinate of an elliptic curve.
- The set of divisors on a hyperelliptic curve with a zero coordinate. For example, the divisors with deg $< g$, is particularly easy to build and detect. In fact, whereas the divisors with some zero coordinates but degree g induce some multiplications by zero, which, can be detected, those with deg $< g$ usually require different routines from those for the generic, most common cases, which have different timings, too.

The probability that random element $\in H$ is $O(q^{-1})$, so the set is small enough to make fulfilling the conditions of step 2 quite easy.

The sets H defined above are clearly preserved by multiplicative divisor randomisation, curve isomorphisms, and field isomorphisms. The side-channel trace correlation may reveal if the guess was correct even in presence of such randomization procedures. In particular, this can affect the random isomorphisms of the form $\phi : (x, y) \mapsto \left(s^{-2}x, s^{-(2g+1)}y\right)$ and the techniques described in § 2.4.2.b.

For elliptic curves, this attack is not too serious, and can be avoided easily, as shown by Smart [87]. Avanzi [7] has shown that the attacks is a much more grave menace to the integrity of hyperelliptic cryptosystems.

In fact, on a genus g curve ($g > 1$), let H be the set of divisors of degree $< g$. After the computation of $E = t \cdot D \in H$, the scalar multiplication algorithm will double it. To do it, the formulae for the most common case will not work - and even if Cantor's algorithm is used to implement the group arithmetic, there will be less reduction steps than in the general case. The consequences are easily detectable differences in power consumption and even timing. The timing differences are the leaked information used in the attack described in [43], which we can thus consider as a *Goubin-type timing attack*.

An approach to thwart Goubin's attack could use very general curve isomorphisms which do not respect the Weierstraß form with b, $A(x) \neq 0$ to randomize also the vanishing coefficients of the divisors: this has the disadvantage of requiring more general and much slower formulae for the operations.

However, as shown by Avanzi, suitable hyperelliptic curve variants of message or scalar blinding can be effective against Goubin's attack. These countermeasures should be completely effective also on hyperelliptic variants of the exceptional procedure attack [35].

2.6 Higher order DSCA

In the DSCA attacks explained so far the attacker monitors leaked signals and calculates the individual statistical properties of the signals at each sample time. In a higher order DPA attack, the attacker calculates joint statistical properties of power consumption at *multiple* sample times *within* the power signals.

More formally, an kth-order DSCA attack is defined as a DSCA that makes use of k different samples in the power consumption signal that correspond to k different intermediate values calculated during the execution of an algorithm.

The attacks described so far are *first order* DSCA attacks. The idea behind higher order DSCA had already been defined in [46]. A description of a second-order DPA can be found in [61].

The approach followed by Rita Mayer-Sommer in [59], and by her defined as a refined SPA, is in fact a higher order DPA: Her attack works by comparing and correlating the characteristics of different parts of the same, single, power trace. By means of this it is possible to locate the reuse of the same operands, or to estimate the differences between different operands when the same instruction is performed more times during the whole cryptographic operation.

We envision here a plausible scenario. Suppose that an implementor chooses a scalar multiplication algorithm based on sliding windows of size, say, 4 and no precaution against SPA has been taken. Then, for a 160-bit exponent one expects about 32 non-zero coefficients and a single trace shows 160 doublings with 32 additions interspersed. At each addition, the intermediate value is added to one of 16 possible precomputed multiples of the base group element D ($\pm D, \pm 3D, \ldots, \pm 15D$). Hence there are $16^{32} = 2^{128}$ different multipliers which can be represented by that trace. However, each precomputed group element is reused on average 2 times. It an attacker can determine *when* a given element is reused, then the number of possible multipliers decreased to 2^{64}. The secret key can then be recovered by a *meet-in-the-middle* or Pollard rho-like approach having square root complexity, with $\propto 2^{32}$ group computations.

If we assume that *reusing* a group element in the computation leads to a correlation in the traces then higher order DSCA can lead to such a drastic key space reduction. On the one hand, we doubt that this is plausible with power analysis, as traces of arithmetic multiplications and addition depend on

both operands involved, and fixing only one should not determine the characteristic of the operation, except for very special values of the operand (0, 1, -1,...). On the other hand, this seems definitely possible with EM emissions recorded in proximity of the device, if they can be used to localize memory accesses!

The idea of attacks based on operand reusage have been proposed first in [94], where they are applied to the behaviour of the multiplier in an implementation of RSA to allow the identification of equal exponent digits.

[71] also proposes such a scenario, which can break the scalar multiplication algorithm proposed in [64] if addition and doubling are distinguishable, and claim a second-order DPA resistant version of that algorithm in [72]. Ahn et al. propose an algorithm [3] whose resistance against second-order DPA is still to be assessed.

Under the assumptions intrinsic to the scenarios just depicted, we recommend the implementor to relocate, at some intervals, the precomputed data in memory. This should be done a few times during a run of the whole algorithm, and in a way that prevents the attacker to determine when the same operand is transferred more times from one memory location to the other. The optimal way of implementing this is not clear.

In order to avoid the detection of reusage of a precomputed point via leaked-emission analysis, it is also advisable to use a redundant representation (projective, jacobian, new-weighted) of the group elements and to randomise the representation of a precomputed point after each time it is used. This is trivial to implement, and relatively inexpensive.

SSCA-countermeasures will also make it more difficult for the attacker to guess which portions of single traces are to be correlated. This shall force him, ultimately, to adopt a brute force strategy, i.e. to try to correlate all possible sub-traces: the amount of possible combinations will increase exponentially with the length of the scalar multiplication and thereby make kth-order attacks computationally infeasible.

2.7 Address-bit DPA

Address-bit differential power analysis (and, similarly, address-bit differential EM emission analysis) does not exploit the knowledge of the internal representation of the operands. Instead, it exploits guesses individual bits of memory addresses or of register numbers in order to detect operand reuse. From this point of view, this attacks is closely related to higher-order DSCA.

In fact, memory addresses are just another type of operand processed by some CPU instructions, and have their internal bit representation. Therefore, accessing different memory locations will produce traces which present less correlation than the traces corresponding to repeated access of the same memory location.

The attack works by sorting all the memory addresses in two sets using one bit of the address as discriminating factor, and then continues further in the same way. Clearly, the attack works better if only a few memory locations are used. The analogous attack using the CPU/coprocessor register numbers, similarly, will be most effective of the amount of registers used is small.

In [62, 63], not only multi-bit DPA was introduced, but also address-bit DPA. Clearly, the two ideas can be combined. These attacks were used also on a DES implementation. May et al. [58] suggest a *random register renaming*, to be implemented in hardware, on the processor called NDISC [57], to foil address-bit DPA attacks. Similar ideas can be used on memory addressing.

Itoh et al. [31] successfully devised an address-bit DPA attack on ECC protocols using elliptic curves in the Montgomery form. In [32] the same authors propose a *randomized addressing countermeasure* along the ideas of [58], but implemented entirely in software. Suppose several operands are

stored in a table contained in the memory of the embedded device. Access is done via a base address and an index. The basic idea, there, is to choose a new integer `r` anew for each session, which has the same bit length as the highest index. Before doing any table access – both during the table initialisation phase and later in the scalar multiplication – the index is `xor`'ed to `r`. This idea just makes it more difficult to detect equal coefficients between different runs of the device, and in particular it is effective against a first order differential power analysis, but it is definitely ineffective against a higher order bit-address DSCA.

Against higher order bit-address DSCA, operand relocation seems the only choice. As we have already remarked in the previous Subsection, it is not clear how to implement this countermeasure in an optimal and effective way.

3 Timing attacks

In our context, timing attacks apply to computations $n \cdot D$, with n fixed, secret, and D an element from an abelian group. The latter can be the rational point group of an elliptic curve, the rational divisor class group of an hyperelliptic curve, the subset of the latter describing a trace-zero variety, etc. The assumption here is that the scalar n is fixed (hence, not even internally randomized!) and that the attacker can choose the input – hence, in a certain sense, timing attacks can be seen as very specific *chosen plaintext leakage analysis attacks*.

The main observation here is that the device/software combination takes different amounts of time to process different inputs. This can be due to the presence of branching/conditional statements in the software, or just because of variable timings of instructions directly in the hardware. An example of this different timing is the implementation of multiprecision modular arithmetic: with all modular reduction algorithm, such as the division with remainder, or Barrett's, Montgomery's or Quisquater's *in their basic forms*, the final result must be checked and, possibly, the modulus has to be subtracted again a few times.

A timing attack is very similar to DSCA, but some details are slightly more complicated, even though this attack was devised earlier. The complication is due to the fact that timing attacks exploit the timing of the *whole* cryptographic operation, and do not observe the timings of the sub-operations in which the cryptographic operations splits. Also, we should not forget that when timing attacks have been introduced, power or EM side channels were not known to the academic world.

Let then $n = (n_r, n_{r-1}, \ldots, n_0)$ in the binary representation. Suppose that the bits $n_r, n_{r-1}, \ldots$, n_{j+1} are known. The attacker wants to find n_j. He proceeds as follows:

1. The attacker first makes a *guess*: $n_j = 0$ or 1.

2. He takes several inputs $D_1, \ldots, D_t$ and simulates the internal state of the device at the moment of the computation of $E_i = \left(\sum_{d=j}^{r} n_d 2^{d-j}\right) D_i$. He then selects a subset of the D_i, according to the following rule: The time necessary to perform the computation of the last addition or doubling giving E_i as a result is strongly biased in one sense or the other. For example he may select those inputs for which a particular modular reduction always (or never) requires a final subtraction. In other words, the timings of the part of the whole computation which processes the bit n_j must be *biased*.

3. Finally, the attacker measures and averages the timings obtained when processing those selected inputs on the whole scalar multiplication. It is expected that the timings of the parts of the scalar multiplications which process the other bits of the scalar are "randomly" distributed.

Whereas the attacker can actually enforce this for the processing of the bits $n_r, \ldots, n_{j+1}$, for the processing of $n_{j-1}, \ldots, n_0$ this must be, essentially, just *hoped*. In other words, the attacker hopes that the bias in time in a given part of the computation depending on the guessed input will result in a bias of the whole computation, if the guess was correct.

4. He compares this average timing with that of a set of random inputs. If these two averages differ significantly, then the probability that the guess of key bit was correct is high, and can even be estimated.

At CARDIS 1998 J.-F. Dhem at al. [23] showed how to make timing attacks work on RSA with Montgomery multiplication. The same idea of course applies also to elliptic curves and hyperelliptic curves. For hyperelliptic curves, the different timings can be caused, for example, by group operations involving a special divisor [43], and thus requiring formulae which are different from those for the most common cosa: as we have seen in Subsection 2.5, this can be understood in the context of Goubin-type timing attacks.

The countermeasures are in principle the same as for differential side channel analysis. Randomised clocking and RPIs are not a countermeasure, because their influence on the total timing can usually be eliminated by averaging enough samples – unless the decisions taken by the hardware to decide how vary the clock or when to add RPIs are deterministically tied to some internal state, but in a secret way.

4 Fault attacks

4.1 Fault Induction

Usually the name *differential fault analysis* is used to denote all the attacks presented in this subsection. However we prefer to use the definition *simple fault analysis* for all attacks that exploit, the observation of the consequences of one (or very few) induced faults to recover the secret key, and the definition *differential fault analysis* to refer to the attacks that extract the bits of the secret key one by one by comparing the faulted results with a correct result.

4.1.1 Simple fault analysis

Attacks based on fault induction essentially force the device to perform erroneous instructions - for example by changing some bits in the internal memory. They have been announced officially in 1996 in a Bellcore press release [79], followed by a paper [13] written by Bellcore researchers, and for this reason these attacks are sometimes called *Bellcore attacks*. See also [52] and [8]. It is known that a simple implementation of RSA which computes separately modulo the two prime moduli and then reconstructs the result using the Chinese Remaindering Theorem can be easily broken by inducing faults: in fact a randomly manipulated result immediately leads to a successful factorization of the RSA modulus. The most obvious way to protect against fault attacks is to check the computation. The device could, for example, repeat it and compare the results. This is of course extremely expensive, either in terms of time, or of hardware space (if we double the hardware and perform the computation twice in parallel). Shamir proposed a more efficient verification procedure [83] for RSA exponentiation, and patented it [84]. This method was later improved by Joye, Paillier and Yen [36].

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). For RSA, this is usually less time-consuming, as the

public exponent is usually chosen to be small. For ECC and HECC this is more or less as expensive as performing the computation twice – unless, also in this case, double hardware is used.

For elliptic curve cryptography, the situation would seem different enough to make a *simple* fault-induction attack more or less meaningless, because from one single, randomly manipulated, wrong result we could naively expect that it is not possible to obtain information on the scalar used in the scalar multiplication. However, there is a very clever, and somewhat plausible, scenario, which can shatter that illusion...

At CRYPTO 2000, Biehl, Meyer, and Müller [10] presented three different types of attacks on ECC that can be used to derive information about the secret key if bit errors can be inserted into the elliptic curve computations in a tamper-proof device. They also estimate the effectiveness of the attacks using a software simulation.

Their methods require *very precise placement and timing* of the faults. Generalizing the approach of [54], their first two DFA techniques depend on the ability to change the coordinates of a point on the curve at the beginning of the scalar multiplication. This can work because the elliptic curve formulae do not use all the coefficients of the equation of the curve to perform their operations: in fact if the original curve is defined, over the field $\mathbb{F}$, by the equation

$$C : Y^2Z + a_1XYZ + a_3YZ^2 = X^3 + a_2X^2Z + a_4XZ^2 + a_6Z^3$$

then it is very likely that a weak curve has an equation differing only in the coefficient a_6. Let the order of the group of $\mathbb{F}$-rational points of

$$C' : Y^2Z + a_1XYZ + a_3YZ^2 = X^3 + a_2X^2Z + a_4XZ^2 + a_6'Z^3$$

be the product of several small prime powers – or at least be divisible by a factor, κ say, of moderate size, such that solving the DLP on C' modulo κ is easy. Such a curve is more easily obtained than a cryptographically strong one, since we have to count points on *less* curves to find a weak one... Now, a_6 is not used in the group formulae, hence if the the base point has been modified, so that it lies on E' then the device will compute the scalar multiplication with the secret scalar on C', and solving the new, easy DLP will return the secret scalar (or the scalar modulo an reasonably sized integer).

Note that if the attacker knows where the y coordinate of the base point will be stored, the attacker can change its contents, for example using a focused ray of elementary particles - but the hardware could use an internally scrambled representation of the bits, so that the logical content of that register will not be known. But the attacker can still choose which point will be input to the device, and the x–coordinate can be assumed to be known. The attacker might know also the *output* point, in all its coordinates. From the latter, it is then possible to determine the curve C' on which the computation took place, to establish the two possibilities for the y coordinate of the modified base point, to count the points on the curve C' and thus, if this is weak, to solve the DLP. Otherwise, he tries again with a different point - hence, with overwhelming probability, with a different curve C'. Trying with at most a few curves will break completely the system.

Here we observe that *all the hyperelliptic curve explicit formulae available do not use all the coefficients of the equation in their computations* - exactly as the elliptic curve formulae. Therefore, a *hyperelliptic curve differential fault analysis attack* is absolutely plausible under the same assumptions made in [10]. The same countermeasures *must* be used also for hyperelliptic curve embedded cryptosystems! The same holds also for the attacks presented next.

4.1.2 Differential fault analysis

We now introduce a class of attacks which induce faults in order to recover the key bits one by one.

Their attack would work on a right-to-left scalar multiplication algorithm as follows:
Let $(n_r, n_{r-1}, \ldots, n_0)_2$ the binary representation of a positive integer n, where n_0 is the least significant bit. Let D be the base divisor, and let the right-to-left scalar multiplication algorithm:

```
H = D; Q = O;
for i = 0 to r do
    if (n_i == 1) then Q = Q + H;
    H = 2 H;
end for;
output Q;
```

Assume that we know the binary length $r+1$ of the unknown multiplier n (note that an attacker can easily guess this length). Denote by $Q(i)$, $H(i)$ the value stored in the variable Q, H in the algorithm description before iteration i. The final result will then be $Q(r+1)$. We first use the tamper-proof device with some input D_E to get the correct result $Q(r+1) = n \cdot D_E$. Then we restart it with input D_E but enforce a random register fault to get a faulty result $\tilde{Q}(r+1)$. Assume that we enforce the register fault in iteration r and that this fault changes the variable H. If the final result is unchanged, then there was no addition in the last iteration and $n_r = 0$, otherwise there was an addition ad $n_r = 1$. Clearly, we can do this for each bit of the scalar.

[10] has also an approach in which faults are introduced during the scalar multiplication in a more sophisticated way. Assume that the attacker can, at any prescribed iteration, flip just one bit of the variable Q, say. (The case where the variable H is modified is handled in a similar way.) Assume also that the scalar is fixed and not randomized, and that we know how the internal variables are represented. Then, essentially, the bits of the key can be recovered in small blocks as follows:

1. Perform a normal scalar multiplication $E = n \cdot D$ with a given input.

2. Repeat the computation of $n \cdot D$, but this time induce a bit flip in a register m steps before the end of the scalar multiplication, giving a result E'. Of course all computations before the fault in the two cases will be equal, and all those involving the processing of the m most significant bits of the unknown scalar n in the faulted computation will have been changed with respect to those of the reference computation – the very first difference consisting in just a single bit.

3. For all possible m bit integers x, "reverse" the correct computation and the faulted one, i.e. determine $E - x \cdot 2^{r+1-m} \cdot D$ and $E' - x \cdot 2^{r+1-m} \cdot D$.

4. If pairs of results which differ in only one bit are found, then the correct and the faulted register values are now determined together with the m most significant bits of the scalar.

5. If one bit of H (which is supposed to contain a copy the input base point D) was flipped, and not one of Q, the computations need to be done not only for all possible combinations of m bits, but also for all possible single bit faults induced in H (this is $O(\log q)$ where q is the cardinality of the used field).

6. Iteration of the above process yields all the bits of the secret scalar m bits at a time.

The attack works essentially changed if the binary representation of the scalar is replaced with any other deterministic recording (NAF, m-ary, w-NAF, etc.).

Biehl et al. indeed have more sophisticated attacks, such as attacks where the faults are induced but it not possible to determine with precision *when* – such attacks are reasonable if clock randomisation or RPIs are implemented on the card.

All such attacks have been originally presented for elliptic curves but are in fact entirely generic and apply to implementations of hyperelliptic curves, trace-zero varieties and other geometric objects. Countermeasures are obvious: checking at regular intervals whether the intermediate result is on the curve, and restarting the computation (possibly from a back-up value) if something has gone wrong; never allowing wrong results to leave the device; randomizing the scalar.

4.2 Adaptive Fault Analysis

Faults can be used to detect *dummy operations* [96, 97]. Dummy operations are used as a simple SSCA countermeasure, however faulting them will not change the final result of a cryptographic operation. *Adaptive fault analysis* can enable an attacker to distinguish the dummy operations from the effective ones, i.e. from those which are essential to the computation of a group operation. We now describe this attack.

Suppose that at a given place of the implementation of the group operations, the doubling routine, say, contains a dummy instruction whereas, at the correspondig place, the addition routine has an effective instruction, i.e. one which belongs to the addition formula. Suppose further that the scalar and its internal representation do not change between different runs of the device. Then very precisely timed faults can be used to distinguish the two types of operation: if the fault does not change the result, then at that moment the device was performing a doubling, otherwise that operation was an addition.

Thus, the key can be recovered in $O(\log n)$ steps, where n is the the secret scalar.

Specific randomization procedures in the scalar multiplication algorithm are necessary, just as for DSCA (or against timing attacks).

If faults or intrusions have been detected, then the computation has to be redone, restarted with the state before the fault, or aborted without output. This can lead to *observable* differences in the behaviour of the system: in [38] the case of an RSA system has been investigated. The observable behaviour can consist in the success/failure to encipher/decipher, but, for example, also in the total timing for the cryptographic operations, or in the power trace/EM emission of the device. In other words, an (apparently) improved cryptosystem may actually leak useful observable information. If this behaviour can be observed for several different faults induced in different moments of a scalar multiplication with fixed scalar, then an attacker may be able to guess the secret key. Also in this case we strongly advice to randomize the scalar.

Some hardware countermeasures against fault induction attacks are rather general countermeasures, independent from the algorithm actually performed. For example, the following approaches, even if introduced for specific cryptosystems, are rather general-purpose:

- Add error-detecting codes to protect the integrity of registers [95];
- Use a self-timed circuit and dual rail-logic to thwart error induction [67]. This is a recent approach whose effectiveness still has to be thoroughly assessed.

There are also more limited approaches: Karri et al. [41, 42] add circuitry to perform, in parallel with the encryption, a reverting of the performed operations: this approach is clearly more oriented towards symmetric systems.

We believe that the combination of error-detection codes, in combination with glue logic, is a sound verification technique for the integrity of internal data, as the glue logic makes in principle impossible for the attacker to modify the content of a register and of the associated error-detection information.

Randomised clocking and RPIs should make it very difficult to correctly place the faults, at least for an attacker with commo equipment.

In theory, however, a much better equipped attacker might use a more sophisticated device that monitors completely the state of the device in order to decide the fault placements in real-time, depending on some observed state, such as a the traces of a very specific instruction sequence. This would probably circumvent randomised clocking, but not completery RPIs.

5 Conclusions

5.1 Suggestions

In order to thwart SSCA, *at least one* of the following countermeasures should be used

- Indistinguishable basic operations, at least by employing atomicity.
- A scalar multiplication algorithm with a fixed sequence of operations, even if addition and doubling are distinguishable.

To prevent DSCA, Goubin-type DSCA, and simple/differential/adaptive fault analysis, we suggest to use *both* the following countermeasure types:

- Randomization of the group, or at least of the base point. (But we do not recommend to randomize the representation of the field.)
- A suitable scalar randomization, but with moderately sized constants. (But not a randomized representation of the same scalar, unless it is a technique resembling the MIST exponentiation algorithm.)

Against fault analysis, checking regularly if the intermediate results are consistent (for example, whether points or divisors are on the curve) is advisable.

Against higher order DSCA based on the EM spectrum and higher order bit-address DSCA, we advise the implementor to make detection of operand reusage difficult. In particular, the already mentioned re-randomisation of precomputed points after their usage and their relocation would be good starting points.

Hardware countermeasures, like buffering of the power source, EM shielding, RPI and randomized clocking, intrusion and fault detection (using for example checksums), are also suggested.

5.2 Directions for future research

Important directions for future research are, in our opinion:

- Research of redundant coordinate systems (see Paragraph 2.4.2.b) for genus 3 curves, to allow divisor randomisation. We recall, however, that curve randomisation is effective and efficient on those curves.
- Development of atomicity (see Subsubsection 2.3.1) for hyperelliptic curve cryptosystems.
- There is need for mathematical models to allow a sound assessment of the solidity of the combination of different countermeasures.
- Develop reasonable models for resistance against higher order DSCA.
- Sound countermeasures against detection of operand reusage under higher order EM emission DSCA (see Subsection 2.6) are still needed, to be used together with scalar multiplication algorithms based on several precomputations in order to speed up computations.
- Montgomery ladders and alternate forms with indistinguishable operations for hyperelliptic curves (see Subsections 2.3.2 and 2.3.3).

5.3 Point of the situation

We are strongly convinced that no perfect protection or countermeasure against side channel, timing analysis or fault induction attacks exists. On one hand, SSCA-countermeasures are usually not effective against DSCA. On the other hand one cannot even speak of protecting an implementation of a cryptographic primitive against DSCA if SSCA-resistance has not been taken into account first. We know that some countermeasures, like the original version of Coron's first countermeasure, were considered potentially weak if used alone. However that countermeasure can be used, in combination with other techniques, to thwart a hyperelliptic version of Goubin's very recent attack, which is a more serious attack that the one against which it had beed originally developed... One should also consider to protect the operations themselves against side channel attacks which are involved in setting up a countermeasure... And so on, and so on... Countermeasures can make the attacker's task harder, and therefore limit the threat to more skilled, more resourceful, better trained adversaries, but not impossible: eventually, it may be only a matter of budget.

Whereas effective countermeasures ("effective" is here intended modulo the cautions just said) have been developed in abundance for elliptic curve cryptosystems, it has been only recently that side channel attacks have been considered on hyperelliptic curve cryptosystems, and corresponding countermeasures were developed and analysed: in particular, research on the latter problems is still in its infancy. However, our (partial) conclusion is that *almost all attacks that can be mounted on an elliptic curve cryptosystem can be mounted also against hyperelliptic curve cryptosystems*. This is no big surprise, since elliptic curves are just particular hyperelliptic curves. The good news is that it seems also that *many countermeasures developed for elliptic curve cryptosystems apply to hyperelliptic systems, too* - we also reviewed how this can be done. It would be interesting to have alternate forms for hyperelliptic curves too, for example.

The existing countermeasures for hyperelliptic cryptosystems suffice, in our opinion, to make implementations of those cryptosystems as secure as implementations of elliptic ones can get.

We are persuaded that only a combination of hardware protections and software countermeasures can effectively lead to "secure" implementations.

Side-channel cryptanalysis is very implementation-specific, but some general approaches and ideas can already be given. There is a need for a sound theoretical framework for side channel cryptanalysis. This should include a thorough physical modelling of the devices and a formally correct, yet concrete, mathematical modelling of the attacks and of the countermeasures. In particular, a study of the combinations of countermeasures seems missing, and all the results so far in that direction are, when available, quite of empiric nature. Results from complexity theory would, in our opinion, play an important role in this reseach. Only this would allow a correct assessment of the potential of attacks and countermeasures. This research should be pursued very actively in the next future.

References

[1] G. B. Agnew, R. C. Mullin, and S. A. Vanstone. An implementation of elliptic curve cryptosystems over $\mathbb{F}_{2^{155}}$. *IEEE Journal on Selected Areas in Communications*, 11(5):804–813, 1993.

[2] D. Agrawal, B. Archambeault, J. R. Rao, and P. Rohatgi. The EM Side-Channel(s). In B. S. Kaliski Jr., Ç. K. Koç, and C. Paar, editors, *CHES 2002*, volume 2523 of *Lecture Notes in Computer Science*, page 29. Springer-Verlag, Berlin, 2003.

[3] M.-K. Ahn, J.-C. Ha, H.-J. Lee, and S.-J. Moon. A Random M-ary Method based Countermeasure against Side Channel Attacks. In *Proceedings of ICCASA-2003*, volume 2668 of *Lectures Notes in Computer Science*, pages 338–347. Springer-Verlag, 2003.

[4] R. Anderson and M. Kuhn. Tamper resistance - a cautionary note. In *Proceedings of the Second Usenix Workshop on Electronic Commerce*, pages 1–11, November 1996.

[5] R. Avanzi and P. Mihăilescu. Generic Efficient Arithmetic Algorithms for PAFFs (Processor Adequate Finite Fields) and related algebraic structures. In *SAC 2003*, volume 3006 of *Lecture Notes in Computer Science*, pages 320–334. Springer-Verlag, Berlin, 2004.

[6] R. M. Avanzi. Aspects of hyperelliptic curves over large prime fields in software implementations. In *Proceedings of CHES 2004*, volume 3156 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 2004.

[7] R. M. Avanzi. Countermeasures against Differential Power Analysis for hyperelliptic curve cryptosystems. In *Cryptographic Hardware and Embedded Systems – CHES 2003*, volume 2779 of *Lecture Notes in Computer Science*, pages 366–381. Springer-Verlag, Berlin, 2004.

[8] F. Bao, R.H Deng, Y. Han, A. Jeng, A.D. Narasimbalu, and T. Ngair. Breaking public key cryptosystems on tamper resistant devices in the presence of transient faults. In *The 1997 Security Protocols Workshop*, Paris, France, 1997.

[9] L. Benini, A. Macii, E. Macii, E. Omerbegovic, M. Poncino, and F. Pro. Energy-Aware Design Techiques for Differential Power Analysis Protection. In *Proceedings of: DAC-40 – ACM/IEEE Design Automation Conference*, pages 36–41. ACM, 2003.

[10] I. Biehl, B. Meyer, and V. Muller. Differential fault attacks on elliptic curve cryptosystems. In *Advances in Cryptology CRYPTO 2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 131–146. Springer-Verlag, Berlin, 2000.

[11] E. Biham and A. Shamir. Differential fault analysis of secret key cryptosystems. In *Crypto'97*, volume 1294 of *Lecture Notes in Computer Science*, pages 513–525. Springer-Verlag, 1998.

[12] O. Billet and M. Joye. The Jacobi model of an elliptic curve and Side-Channel Analysis. In *Applicable Algebra, Algebraic Algorithms and Error-Correcting Codes – AAECC 2003*, volume 2643 of *Lecture Notes in Computer Science*, pages 34–42. Springer-Verlag, Berlin, 2003.

[13] D. Boneh, R. DeMillo, and R. Lipton. On the importance of checking cryptographic protocols faults. In *Proceedings of Eurocrypt '1997*, volume 1233 of *Lecture Notes in Computer Science*, pages 37–51. Springer-Verlag, Berlin, 1997.

[14] É. Brier, I. Déchène, and M. Joye. Unified point addition formulæ for elliptic curve cryptosystems. In N. Nedjah and L. de Macedo, editors, *Embedded Cryptographic Hardware: Methodolgies & Architectures*. Nova Science Publishers, 2004.

[15] É. Brier and M. Joye. Weierstraß elliptic curves and side channels attacks. In D. Naccache and P. Paillier, editors, *Public Key Cryptography – PKC 2002*, volume 2274 of *Lecture Notes in Computer Science*, pages 335–345. Springer-Verlag, 2002.

[16] B. Chevallier-Mames, M. Ciet, and M. Joye. Low-cost solutions for preventing simple Side-Channel Analysis: Side-Channel Atomicity. *IEEE Trans. on Computers*, 53:760–768, 2004. also available from Cryptology ePrint Archive.

[17] M. Ciet. *Aspects of Fast and Secure Arithmetics for Elliptic Curve Cryptography*. PhD thesis, Université Catholique de Louvain, June 2003.

[18] M. Ciet, J.-J. Quisquater, and F. Sica. Preventing differential analysis in GLV elliptic curve scalar multiplication. In *Cryptographic Hardware and Embedded Systems – CHES 2002*, volume 2523 of *Lecture Notes in Computer Science*, pages 540–550. Springer-Verlag, Berlin, 2002.

[19] C. Clavier, J.S. Coron, and N. Dabbous. Differential Power Analysis in the Presence of Hardware Countermeasures. In *Proceedings of: Cryptographic Hardware and Embedded Systems - CHES 2000*, volume 1965 of *Lecture Notes in Computer Science*, pages 252–263. Springer-Verlag, 2000.

[20] H. Cohen, A. Miyaji, and T. Ono. Efficient elliptic curve exponentiation using mixed coordinates. In *Proceedings of Asiacrypt'98*, volume 1514 of *Lecture Notes in Computer Science*, pages 51–65. Springer-Verlag, Berlin, 1998.

[21] J.-S. Coron. Resistance against differential power analysis for elliptic curve cryptosystems. In *Cryptographic Hardware and Embedded Systems CHES 1999*, volume 1717 of *Lecture Notes in Computer Science*, pages 392–302. Springer-Verlag, Berlin, 1999.

[22] J.S. Coron, P. Kocher, and D. Naccache. Statistics and Secret Leakage. In *Financial Cryptography '00*, 2000.

[23] J.-F. Dhem, F. Koeune, P.-A. Leroux, P. Mestré, J.-J. Quisquater, and J.-L. Willems. A practical implementation of the timing attack. In J.-J. Quisquater and B. Schneier, editors, *Third smart card research and advanced application conference – CARDIS 98*, volume 1820 of *Lecture Notes in Computer Science*, pages 167–182. Springer-Verlag, Berlin, 1998.

[24] W. Fischer, C. Giraud, E. Woodward Knudsen, and J.P. Seifert. Parallel scalar multiplication on general elliptic curves over $\mathbb{F}_p$ hedged against non-differential side-channel attacks. preprint, January 2002.

[25] R. P. Gallant, J. L. Lambert, and S. A. Vanstone. Faster point multiplication on elliptic curves with efficient endomorphisms. In *Advances in Cryptology – Crypto'2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 190–200. Springer-Verlag, Berlin, 2001.

[26] K. Gandolfi, C. Mourtel, and F. Olivier. Electromagnetic analysis: concrete results. In *Workshop on Cryptographic Hardware and Embedded Systems CHES 2001*, volume 2162 of *Lecture Notes in Computer Science*, pages 251–261. Springer-Verlag, Berlin, 2001.

[27] L. Goubin. A refined power analysis attack on elliptic curve cryptosystems. In *Public Key Cryptography – PKC 2003*, volume 2567 of *Lecture Notes in Computer Science*, pages 199–210. Springer-Verlag, Berlin, 2003.

[28] J. Ha and S. Moon. Randomized signed-scalar multiplication of ECC to resist power attacks. In *Cryptographic Hardware and Embedded Systems – CHES 2002*, volume 2523 of *Lecture Notes in Computer Science*, pages 551–563. Springer-Verlag, Berlin, 2002.

[29] D. Hankerson, J. López, and A. J. Menezes. Software implementation of elliptic curve cryptography over binary fields. In *Cryptographic Hardware and Embedded Systems, CHES (Worcester, 2000)*, volume 1965 of *Lecture Notes in Computer Science*, pages 1–24. Springer-Verlag, Berlin, 2000.

[30] Y. Hitchcock and P. Montague. A new elliptic curve scalar multiplication algorithm to resist simple power analysis. In *Information Security and Privacy – ACISP 2002*, volume 2384 of *Lecture Notes in Computer Science*, pages 214–225. Springer-Verlag, Berlin, 2002.

[31] K. Itoh, T. Izu, and M. Takenaka. Address-Bit Differential Power Analysis of Cryptographic Schemes OK-ECDH and OK-ECDSA. In *Cryptographic Hardware and Embedded Systems – CHES 2002*, volume 2523 of *Lecture Notes in Computer Science*, pages 129–143. Springer-Verlag, 2002.

[32] K. Itoh, T. Izu, and M. Takenaka. A Practical Countermeasure against Address-Bit Differential Power Analysis. In *Proceedings of CHES 2003*, volume 2779 of *Lecture Notes in Computer Science*, pages 382–396. Springer-Verlag, 2003.

[33] K. Itoh, J. Yajima, M. Takenaka, and N. Torii. DPA countermeasures by improving the window method. In *Cryptographic Hardware and Embedded Systems – CHES 2002*, volume 2523 of *Lecture Notes in Computer Science*, pages 303–317. Springer-Verlag, 2003.

[34] T. Izu and T. Takagi. A fast parallel elliptic curve multiplication resistant against Side-Channel Attacks. In *Public Key Cryptography – PKC 2002*, volume 2274 of *Lecture Notes in Comput. Sci*, pages 280–296. Springer-Verlag, Berlin, 2002.

[35] T. Izu and T. Takagi. Exceptional procedure attack on elliptic curve cryptosystems. In Y. G. Desmedt, editor, *Public Key Cryptography – PKC 2003*, volume 2567 of *Lecture Notes in Comput. Sci*, pages 224–239. Springer-Verlag, Berlin, 2003.

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

[37] M. Joye and J.-J. Quisquater. Hessian elliptic curves and side-channel attacks. In *Cryptographic Hardware and Embedded Systems – CHES 2001*, volume 2162 of *Lecture Notes in Computer Science*, page 402. Springer-Verlag, Berlin, 2001.

[38] M. Joye, J.-J. Quisquater, S.-M. Yen, and M. Yung. Observability analysis – detecting when improved cryptosystems fail. In *Topics in Cryptology – CT-RSA 2002*, volume 2271 of *Lecture Notes in Computer Science*, pages 17–29. Springer-Verlag, Berlin, 2002.

[39] M. Joye and C. Tymen. Protections Against Differential Analysis for Elliptic Curve Cryptography – an Algebraic aAproach. In *Cryptographic Hardware and Embedded Systems CHES 2001*, volume 2162 of *Lecture Notes in Computer Science*, pages 377–390. Springer-Verlag, Berlin, 2002.

[40] M. Joye and S.-M. Yen. The Montgomery powering ladder. In *Cryptographic Hardware and Embedded Systems – CHES 2002*, volume 2523 of *Lecture Notes in Computer Science*, pages 291–302. Springer-Verlag, 2003.

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

[42] R. Karri, K. Wu, P. Mishra, and Y. Kim. Fault-based side-channel cryptanalysis tolerant Rijndael symmetric block cipher architecture. In *IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT'01)*. IEEE, 2001.

[43] M. Katagi, I. Kitamura, T. Akishita, and T. Takagi. Novel efficient implementations of hyperelliptic curve cryptosystems using degenerate divisors. In *Workshop on Information Security Applications – WISA 2004*, volume 3325 of *Lecture Notes in Computer Science*, pages 347–361. Springer-Verlag, Berlin, 2004. Also available from Cryptology ePrint Archive.

[44] P. Kocher. Timings attacks on implementations of Diffie–Hellman, RSA, DSS and other systems. In *Advances in Cryptology – Crypto 1996*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113. Springer-Verlag, Berlin, 1996.

[45] P. Kocher, J. Jaffe, and B. Jun. A computational introduction to number theory and algebra.

[46] P. Kocher, J. Jaffe, and B. Jun. Differential power analysis. In *Advances in Cryptology – Crypto 1999*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer-Verlag, Berlin, 1999.

[47] O. Kömmerling and M. G. Kuhn. Design principles for tamper-resistant smartcard processors. In *Proceedings of the Usenix Workshop on Smartcard Technology, Chicago, 10–11 May*, pages 9–20, 1999.

[48] T. Lange. Efficient arithmetic on genus 2 hyperelliptic curves over finite fields via explicit formulae. preprint, 2002.

[49] T. Lange. Inversion-free arithmetic on genus 2 hyperelliptic curves. preprint, 2002.

[50] T. Lange. Weighted coordinates on genus 2 hyperelliptic curves. preprint, 2002.

[51] T. Lange. Formulae for arithmetic on genus 2 hyperelliptic curves, 2004. to appear in J. AAECC.

[52] A.K. Lenstra. Memo on RSA signature generation in the presence of faults. Manuscript, September 1996.

[53] P.-Y. Liardet and N. P. Smart. Preventing SPA/DPA in ECC systems using the Jacobi form. In *Cryptographic Hardware and Embedded Systems CHES 2001*, volume 2162 of *Lecture Notes in Computer Science*, page 391. Springer-Verlag, Berlin, 2001.

[54] C. H. Lim and P. J. Lee. A key recovery attack on discrete log-based schemes using a prime order subgroup. In *Advances in Cryptology – Crypto 97*, number 1294 in Lecture Notes in Computer Science, pages 249–263. Springer-Verlag, 1997.

[55] J. López and R. Dahab. Fast multiplication on elliptic curves over GF(2^m) without precomputation. In *Cryptographic Hardware and Embedded Systems CHES (Worcester, 1999)*, volume 1717 of *Lecture Notes in Computer Science*, pages 316–327. Springer-Verlag, Berlin, 1999.

[56] R.G. Lyons. *Understanding Digital Signal Processing*. Addison Wesley Longman, 1997.

[57] D. May, H.L. Muller, and N.P. Smart. Non-Deterministic Processors. In *Information Security and Privacy, 6th Australasian Conference, – ACISP 2001*, volume 2119 of *Lecture Notes in Computer Science*, pages 115–129. Springer-Verlag, 2001.

[58] D. May, H.L. Muller, and N.P. Smart. Random Register Renaming to Foil DPA. In *Cryptographic Hardware and Embedded Systems CHES 2001*, volume 2162 of *Lecture Notes in Computer Science*, pages 28–38, Berlin, 2002. Springer-Verlag.

[59] R. Mayer-Sommer. Smartly Analyzing the Simplicity and the Power of Simple Power Analysis on Smart Cards. In *CHES 2000*, volume 1965 of *Lecture Notes in Computer Science*, pages 78–92. Springer-Verlag, Berlin, 2000.

[60] T. Messerges, E. Dabbish, and R. Sloan. Examining smart-card security under the thread of power analysis attacks. *IEEE Trans. on Computers*, 51(5):541–552, 2002.

[61] T. S. Messerges. Using second order power analysis to attack dpa resistant software. In *CHES 2000*, volume 1965 of *Lecture Notes in Computer Science*, pages 238–251. Springer-Verlag, Berlin, 2000.

[62] T. S. Messerges, E. A. Dabbish, and R. H. Sloan. Investigations of power analysis attacks on smart cards. In *Workshop on Cryptographic Hardware and Embedded Systems (CHES) 1999*, 1999.

[63] T. S. Messerges, E. A. Dabbish, and R. H. Sloan. Power analysis attacks of modular exponentiation in smart cards. In *Usenix*, 1999.

[64] B. Möller. Securing elliptic curve point multiplication against Side-Channel Attacks. In *Information Security – ISC 2001*, volume 2200 of *Lecture Notes in Computer Science*, pages 324–334. Springer-Verlag, 2001. see also [65].

[65] B. Möller. Securing elliptic curve point multiplication against Side-Channel Attacks. Technical report, TU Darmstadt, 2001. Addendum "Efficiency Improvement" added 2001-08-27/2001-08-29. Errata added 2001-12-21.

[66] P. L. Montgomery. Speeding the Pollard and elliptic curve methods of factorization. *Math. Comp.*, 48(177):243–264, 1987.

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

[68] D. Mumford. *Tata lectures on theta II*. Birkhäuser, 1984.

[69] K. Okeya, H. Kurumatani, and K. Sakurai. Elliptic curves with the Montgomery-form and their cryptographic applications. In *Public Key Cryptography – PKC 2000*, volume 1751 of *Lecture Notes in Computer Science*, pages 238–257. Springer-Verlag, Berlin, 2000.

[70] K. Okeya and K. Sakurai. Power analysis breaks elliptic curve cryptosystems even secure against the timing attack. In *Progress in Cryptology – Indocrypt 2000*, volume 1977 of *Lecture Notes in Computer Science*, pages 178–190. Springer-Verlag, Berlin, 2000.

[71] K. Okeya and K. Sakurai. A Second-Order DPA Attack Breaks a Window-Method Based Countermeasure against Side Channel Attacks. In *Information Security Conference (ISC 2002),*, volume 2433 of *Lecture Notes in Computer Science*, pages 389–401. Springer-Verlag, Berlin, 2002.

[72] K. Okeya and K. Sakurai. On insecurity of the Side Channel Attack countermeasure using Addition-Subtraction Chains under Distinguishability between Addition and Doubling. In *Information Security and Privacy – ACISP 2002*, volume 2384 of *Lecture Notes in Computer Science*, pages 420–435. Springer-Verlag, Berlin, 2002.

[73] K. Okeya and K. Sakurai. A simple power attack on a randomized addition-subtraction chains method for elliptic curve cryptosystems. *IEICE Transactions*, E86-A:1171–1180, 2003.

[74] E. Oswald and M. Aigner. Randomized addition-subtraction chains as a countermeasure against power attacks. In *Cryptographic Hardware and Embedded Systems – CHES 2001*, volume 2162 of *Lecture Notes in Computer Science*, pages 39–50. Springer-Verlag, Berlin, 2001.

[75] J. Pelzl, T. Wollinger, J. Guajardo, and C. Paar. Hyperelliptic Curve Cryptosystems: Closing the Performance Gap to Elliptic Curves. In *Cryptographic Hardware and Embedded Systems – CHES 2003*, volume 2779 of *Lecture Notes in Computer Science*, pages 351–365. Springer-Verlag, 2003.

[76] J. Pelzl, T. Wollinger, and C. Paar. High Performance Arithmetic for Hyperelliptic Curve Cryptosystems of Genus Two. In *International Conference on Information Technology: Coding and Computing – ITCC 2004*, 2004.

[77] J.-J. Quisquater and D. Samylde. A new tool for non-intrusive analysis of smart cards based on electro-magnetic emissions, the SEMA and DEMA methods. Presented at the rump session of Eurocrypt 2000.

[78] P. Rakers, L. Connell, and T. Colins. Secure Contactless Smart-card ASIC with DPA Protection. In *Proceedings of the IEEE 2000 Custom Integrated Circuits Conference*, pages 239–242, May 2000.

[79] Bellcore Press Release. New threat model breaks crypto codes, 1996.

[80] S.Chari, C. Jutla, J. Rao, and P. Rohatgi. Towards sound approaches to counteract power-analysis attacks. In *Advances in Cryptology CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 398–412. Springer-Verlag, Berlin, 1999.

[81] SEPI. Primo simposio nazionale sulla sicurezza elettromagnetica nella protezione dell'informazione, Rome, 1988.

[82] SEPI. Symposium on electromagnetic security for information protection, Rome, 1991.

[83] A. Shamir. How to check modular exponentiation. Presented at the rump session of Eurocrypt '97.

[84] A. Shamir. Method and apparatus for protecting public key schemes from timing and fault attacks. United States Patent 5991415, 1999.

[85] A. Shamir. Protecting smart cards from passive power analysis with detached power supplies. In *Workshop on Cryptographic Hardware and Embedded Systems (CHES) 2000*, volume 1965 of *Lecture Notes in Computer Science*, pages 71–77, Berlin, 2000. Springer-Verlag.

[86] F. Sica, M. Ciet, and J.-J. Quisquater. Analysis of the Gallant-Lambert-Vanstone method based on efficient endomorphisms: Elliptic and hyperelliptic curves. In H. Heys and K. Nyberg, editors, *Selected Areas in Cryptography – SAC 2002*, volume 2595 of *Lecture Notes in Computer Science*, pages 21–36. Springer-Verlag, 2002.

[87] N. P. Smart. An analysis of Goubin's refined power analysis attack. In *Cryptographic Hardware and Embedded Systems – CHES 2003*, volume 2779 of *Lecture Notes in Computer Science*, pages 281–290. Springer-Verlag, Berlin, 2003.

[88] K. Tiri, M. Akmal, and I. Verbauwhede. A Dynamic and Differential CMOS Logic with Signal Independent Power Consumption to Withstand Differential Power Analysis on Smart Cards. In *28th European Solid-State Circuits Conference (ESSCIRC 2002)*.

[89] C. D. Walter. MIST: An efficient, randomized exponentiation algorithm for resisting power analysis. In *Topics in Cryptology – CT-RSA 2002*, volume 2271 of *Lecture Notes in Computer Science*, pages 53–66. Springer-Verlag, Berlin, 2001.

[90] C. D. Walter. Breaking the Liardet–Smart randomized exponentiation algorithm. In *Smart Card Research and Advanced Applications – CARDIS '02*, pages 59–68. Usenix Association, 2002.

[91] C. D. Walter. Seeing through MIST given a small fraction of an RSA private key. In *Topics in Cryptology – CT-RSA 2003*, volume 2612 of *Lecture Notes in Computer Science*, pages 391–402. Springer-Verlag, Berlin, 2002.

[92] C. D. Walter. Some security aspects of the MIST randomized exponentiation algorithm. In *Cryptographic Hardware and Embedded Systems – CHES 2002*, volume 2523 of *Lecture Notes in Computer Science*, pages 276–290, Berlin, 2002. Springer-Verlag.

[93] C. D. Walter. Security constraints on the oswald–aigner exponentiation algorithm. preprint, 2003.

[94] C.D. Walter. Sliding windows succumbs to Big Mac attack. In *Proceedings of: Cryptographic Hardware and Embedded Systems – CHES 2001*, volume 2162 of *Lecture Notes in Computer Science*, pages 286–299. Springer-Verlag, 2001.

[95] 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. In *IEEE international symposium on circuits and systems*. IEEE, 2000.

[96] S.-M. Yen and M. Joye. Checking before output may not be enough against fault-based cryptanalysis. *IEEE Trans. on Computers*, 49(9):967–970, 2000.

[97] S.-M. Yen, S. Kim, S. Lim, and S. Moon. A countermeasure against one physical cryptanalysis may benefit another attack. In *ICICS 2001*, volume 2288 of *Lecture Notes in Computer Science*, pages 414–427. Springer-Verlag, 2002.