Horizontal Collision Correlation Attack on Elliptic Curves

– Extended Version* –

Aurélie Bauer, Eliane Jaulmes, Emmanuel Prouff, Jean-René Reinhard, and Justine Wild

ANSSI, 51, Bd de la Tour-Maubourg, 75700 Paris 07 SP, France
firstname.name@ssi.gouv.fr

Abstract. Elliptic curves based algorithms are nowadays widely spread among embedded systems. They indeed have the double advantage of providing efficient implementations with short certificates and of being relatively easy to secure against side-channel attacks. As a matter of fact, when an algorithm with constant execution flow is implemented together with randomization techniques, the obtained design usually thwarts classical side-channel attacks while keeping good performances. Recently, a new technique that makes randomization ineffective, has been successfully applied in the context of `RSA` implementations. This method, related to a so-called *horizontal modus operandi*, introduced by Walter in 2001, turns out to be very powerful since it only requires leakages on a single algorithm execution. In this paper, we combine such kind of techniques together with the *collision correlation* analysis, introduced at CHES 2010 by Moradi *et al.*, to propose a new attack on elliptic curves atomic implementations (or unified formulas) with input randomization. We show how it may be applied against several state-of-the art implementations, including those of Chevallier-Mames *et al.*, of Longa and of Giraud-Verneuil and also Bernstein and Lange for unified Edward's formulas. Finally, we provide simulation results for several sizes of elliptic curves on different hardware architectures. These results, which turn out to be the very first horizontal attacks on elliptic curves, open new perspectives in securing such implementations. Indeed, this paper shows that two of the main existing countermeasures for elliptic curve implementations become irrelevant when going from vertical to horizontal analysis.

Keywords: side-channel analysis, elliptic curves implementations, ECDSA, horizontal attacks, collision attacks.

1 Introduction

Elliptic Curves Cryptosystems (`ECC`) that have been introduced by N. Koblitz [37] and V. Miller [47], are based on the notable *discrete logarithm problem*,

* The short version of this paper has been published in [8].

which has been thoroughly studied in the literature and is supposed to be a hard mathematical problem. The main benefit in elliptic curves based algorithms is the size of the keys. Indeed, for the same level of security, the schemes require keys that are far smaller than those involved in classical public-key cryptosystems. The success of ECC led to a wide variety of applications in our daily life and they are now implemented on lots of embedded devices: smart-cards, micro-controller, and so on. Such devices are small, widespread and in the hands of end-users. Thus the range of threats they are confronted to is considerably wider than in the classical situation. In particular, physical attacks are taken into account when assessing the security of the application implementation (*e.g.* the PACE protocol in e-passports [33]) and countermeasures are implemented alongside the algorithms.

A physical attack may belong to one of the two following families: *perturbation analysis* or *observation analysis*. The first one tends to modify the cryptosystem processing with laser beams, clock jitter or voltage perturbation. Such attacks can be thwarted by monitoring the device environment with captors and by verifying the computations before returning the output. The second kind of attacks consists in measuring a physical information, such as the power consumption or the electro-magnetic emanation, during sensitive computations. Inside this latter area we can distinguish, what we call *simple attacks*, that directly deduces the value of the secret from one or a small number of observation(s) (*e.g. Simple Power Analysis* [40]) and *advanced attacks* involving a large number of observations and exploiting them through statistics (*e.g. Differential Power Analysis* [41] or *Correlation Power Analysis* [15]). Such attacks require the use of a statistical tool, also known as a *distinguisher*, together with a *leakage model* to compare hypotheses with real traces (each one related to known or chosen inputs). The latter constraint may however be relaxed thanks to the so-called *collision attacks* [54] which aim at detecting the occurrences of colliding values during a computation, that can be linked to the secret [12, 21, 50, 51]. In order to counteract all those attacks, randomization techniques can be implemented (*e.g.* scalar/message blinding for ECC [25]). The recent introduction of the so-called *horizontal* side-channel technique by Clavier *et al.* in [20] seems to have set up a new deal. This method, which is inspired by Walter's work [58], takes its advantage in requiring a unique power trace, thus making classical randomization techniques ineffective. Up to now, it has been applied successfully on RSA implementations and we show in this paper that it can be combined with collision correlation analysis to provide efficient attack on elliptic curves protected implementations.

Core idea. In the context of embedded security, most ECC protocols (*e.g.* ECDSA [2] or ECDH [3]) use a short term secret that changes at each protocol iteration. In this particular setting, advanced side-channel attacks, which require several executions of the algorithm with the same secret, are ineffective. As a consequence, only protection against S-PA is usually needed, that can be done thanks to the popular *atomicity* principle [17, 29, 44]. Up to now, this technique is considered

as achieving the best security/efficiency trade-off to protect against side-channel analysis. In this paper, we provide a new side-channel attack, called *horizontal collision correlation analysis* that defeats such protected ECC implementations. In particular, implementations using point/scalar randomization combined with atomicity are not secure, contrary to what was thought up to now. Moreover in order to complete our study, we also investigate the case of unified formulas[1]. Indeed, we show that our horizontal collision correlation attack allows to distinguish, with a single leakage trace, a doubling operation from an addition one. This technique, which allows to eventually recover the secret scalar, is applied to three different atomic formulae on elliptic curves, namely those proposed by Chevallier-Mames *et al.* in [17], by Longa in [44], by Giraud and Verneuil in [29].

Paper Organisation. The paper is organized as follows. First, Section 2 introduces a framework enabling to formally study the resistance of an implementation against side-channel attacks in both Horizontal and Vertical modes. Section 3 recalls some basics about ECC in a side-channel attacks context. Then, under the assumption that one can distinguish common operands in modular multiplications, the outlines of our new *horizontal collision correlation* attack are presented in Section 4. After a theoretical analysis explaining how to practically deal with the distinguishability assumption, we provide in Section 5 experimental results for 160, 256 and 384-bit-size curves working with 8 or 32-bit registers. These results show that the attack success rate stays high even when significant noise is added to the leakage.

2 A Comprehensive Study of Side-Channel Analyses

In the following, a general framework is introduced which enables to describe most of the existing attacks in a similar way and to identify their core differences (actually the leakage pre-treatment, the leakage model and the statistical test).

2.1 A General Framework for Simple and Advanced Side-Channel Analyses

Theoretically modelling a side-channel analysis obviously requires some prior statistical notions.

Notations. A realization of a random variable X is referred to as the corresponding lower-case letter x. A *sample* of observations of X is denoted by (x) or by $(x_i)_i$ when an indexation is needed. In this case, the global event is sum up as $(x) \hookleftarrow X$. The i^{th} coordinate of a variable X (resp. x), viewed as a vector, is denoted by $X[i]$ (resp. $x[i]$). As usual, the notation $\mathbb{E}[X]$ refers to the mean of X.

[1] Among the unified formulas, we especially focus on the Edward's ones in [9] introduced by Bernstein and Lange since they lead to efficient doubling and addition computations compared to the Weierstrass case [16].

For clarity reasons we sometimes use the notation $\mathbb{E}_X[\cdot]$ to enlighten the variable over which the mean is computed. The variance of X is denoted by $\mathrm{var}(X)$.

Attacks presented in this paper involve the *linear correlation coefficient* which measures the linear interdependence between two variables X and Y. It is defined as $\rho(X,Y) = \frac{\mathrm{cov}(X,Y)}{\sigma_X \sigma_Y}$, where $\mathrm{cov}(X,Y)$, called *covariance between X and Y*, equals $\mathbb{E}[(X - \mathbb{E}[X]).(Y - \mathbb{E}[Y])] = \mathbb{E}[XY] - \mathbb{E}[X]\mathbb{E}[Y]$ and where σ_X and σ_Y respectively denotes the standard deviation of X and Y. The linear correlation coefficient can be approximated from realizations samples $(x_i)_{1 \leq i \leq n}$ and $(y_i)_{1 \leq i \leq n}$ of X and Y respectively. For this approximation, the following so-called *Pearson's coefficient* is usually involved:

$$\hat{\rho}(X,Y) = \frac{n \sum_i x_i y_i - \sum_i x_i \sum_j y_j}{\sqrt{n \sum_i x_i^2 - \left(\sum_i x_i\right)^2} \sqrt{n \sum_j y_j^2 - \left(\sum_j y_j\right)^2}} \ . \tag{1}$$

Attack Context. In the subsequent descriptions of side-channel analyses, an algorithm $\mathcal{A}$ is modelled by a sequence of *elementary calculations* $(\mathsf{C}_i)_i$ that are Turing machines augmented with a common random access memory (see [46] for more details about this model). Each elementary calculation C_i reads its input X_i in this memory and updates it with its output $O_i = \mathsf{C}_i(X_i)$. All the attacks below target a variable $O(\mathbf{s}, X)$ defined as the output of a specific computation (*e.g.* a multiplication) performed by the device and parametrized by a secret sub-part $\mathbf{s}$ and a public variable[2] X. In the following, we shall use O instead of $O(\mathbf{s}, X)$ if there is no ambiguity on $\mathbf{s}$ and X.

To recover information on $\mathbf{s}$, the attacks are performed on a sample of observations related to the processing of O by the device. Each of those observations, such as power consumption, electromagnetic emanations, and so on, is usually composed of several physical measurements corresponding to leakages at different times t_i. It can hence be viewed as a realization of a multivariate random variable $\boldsymbol{L}$ whose coordinates $\boldsymbol{L}[i]$ satisfy:

$$\boldsymbol{L}[i] = \varphi_i\big(O\big) + B_i \ , \tag{2}$$

where φ_i only depends on the device behaviour at time t_i during the processing of O and B_i is an independent Gaussian noise with zero mean and standard deviation σ_i. The function φ_i is *a priori* unknown. The index i will be sometimes omitted. In this case, it is assumed that the same function is associated to all the time indices. See Figure 1 to illustrate the notations and Appendix A for an extension of the definitions and notations to higher-order contexts.

An SCA is based on the *Hypothesis Testing principle* [38]. To make this test, a set of *prediction values* $\boldsymbol{h}_j$ are deduced from each hypothesis $\hat{\mathbf{s}}$ on $\mathbf{s}$ and from the sample of implementation inputs (x_j) corresponding to the observations. This step involves a *leakage model function* $\mathbf{m}$ that must have been priorly chosen

[2] We shall sometimes need to consider the known value as a pair of variables: in this case we will use the notation (X, Y) instead of X.

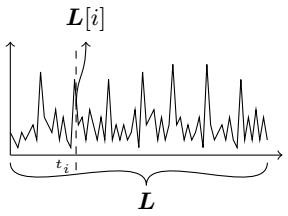

Fig. 1. Power consumption of the processing of O

by the attacker (for instance based on its knowledge on the attacked device architecture). With this model function, the prediction values $\boldsymbol{h}_j$ are built s.t. $\boldsymbol{h}_j = \mathbf{m}(O(\hat{\mathbf{s}}, x_j))$. Eventually, the adversary uses a distinguisher ρ to compare the $\boldsymbol{h}_j$ with the observations $\boldsymbol{l}_j \hookleftarrow \boldsymbol{L}|X = x_j$. This results in a so-called *score value*, $\Delta[\hat{\mathbf{s}}]$ for the hypothesis $\hat{\mathbf{s}}$. The tuple of scores $(\Delta[\hat{\mathbf{s}}])_{\hat{\mathbf{s}}}$ is called *scores vector*.

The overall set of SCA is usually split in two subclasses. The first one, called *simple Side-Channel Analysis*, contains all attacks where observations only need to be done on a single value of the public input parameter (this implies that all the x_j are equal to a same value, say x). This set contains `S-PA` [40], `S-EMA` [27, 53] or `S-TA` (Timing Analysis) [40]. The second subclass, called *advanced Side-Channel Analysis*, includes attacks where observations of the targeted internal processing must be done for several public input parameters. In particular, it contains *univariate SCA* attacks such as `D-PA` [42], `C-PA` [15] or `MI-PA` [28] and *multivariate SCA* attacks such as `HO-D-PA` [42, 52] or `HO-MI-PA` [6]. We give hereafter a more formal description of those two subclasses.

Simple SCA. The class of simple SCA includes all Vertical or Horizontal SCA where the adversary makes observations on a single input. Table 1 provides a description of a simple Side-Channel Analysis[3].

1. Choose a value x for X.
2. Measure a sample $(\boldsymbol{l}_j)_j \hookleftarrow (\boldsymbol{L}|X = x)$ of N leakages.
3. Select a distinguisher Δ and choose a model function $\mathbf{m}$.
4. For each hypothesis $\hat{\mathbf{s}}$ on $\mathbf{s}$, compute $\boldsymbol{h} = \mathbf{m}(O(\hat{\mathbf{s}}, x))$.
5. For each $\hat{\mathbf{s}}$, compute $\Delta[\hat{\mathbf{s}}] = \Delta\big[(\boldsymbol{l}_j)_j, \boldsymbol{h}\big]$.
6. Deduce from $\Delta[\cdot]$ information on $\mathbf{s}$.

Table 1. Simple Side-Channel Analysis

[3] In contexts where the adversary is not allowed to choose the algorithm input but knows it, the first step just aims at fixing the input value for the rest of the attack.

Remark 1. In theory, simple SCA may be conducted with a single observation. In practice however, it is often necessary to use several observations of the processing for the same variable x in order to reduce the noise impact. In this case, the statistical distinguisher ρ may for instance involve a preliminary step consisting in the averaging of the observations sample.

Example 1 (S-PA against an RSA implementation based on the left-to-right Square and Multiply *algorithm).* In this case, the target processing is the entire computation $O = X^{\mathbf{s}} \bmod n$. For a constant message $x \hookleftarrow X$, the adversary starts by getting a sample of observations $(\boldsymbol{l}_j) \hookleftarrow \boldsymbol{L}|X = x$. Then, ranging i from the index of the key most significant bit to 0, he makes an hypothesis $\hat{\mathbf{s}}[i]$. From the model, he hence gets, for each index i, a pattern $p_{\hat{\mathbf{s}}[i]}$ that is assumed to correspond to the power consumption profile of either a squaring plus a multiplication if $\hat{\mathbf{s}}[i]$ equals 1 or a squaring alone if $\hat{\mathbf{s}}[i]$ equals 0. Eventually, the adversary chooses the *Euclidean distance* as distinguisher Δ and computes it between $\boldsymbol{h} = p_{\hat{\mathbf{s}}[i]}$ and the part of the mean vector $\frac{1}{N}\sum_j \boldsymbol{l}_j$ corresponding to the i^{th} step of the exponentiation. The hypothesis $\hat{\mathbf{s}}[i]$ minimizing the distance is assumed to be the most likely one.

Advanced SCA. All the attacks where observations must involve several inputs belong to the *advanced SCA* category. This kind of attacks follows the outlines given in Table 2.

1. Get N measurements $(\boldsymbol{l}_j, x_j)_j \hookleftarrow (\boldsymbol{L}, X)$.
2. Select a distinguisher Δ and choose a model function $\mathbf{m}$.
3. For each hypothesis $\hat{\mathbf{s}}$ on $\mathbf{s}$ build a set of predictions $\boldsymbol{h}_j$ such that $\boldsymbol{h}_j = \mathbf{m}(o(\hat{\mathbf{s}}, x_j))$.
4. For each $\hat{\mathbf{s}}$, compute $\Delta[\hat{\mathbf{s}}] = \Delta\big[(\boldsymbol{h}_j)_j, (\boldsymbol{l}_j)_j\big]$
5. Deduce from $\Delta[\cdot]$ information on $\mathbf{s}$.

Table 2. Advanced Side-Channel Analysis (A-SCA)

Remark 2. Depending on the statistical treatment processed by the distinguisher, the latter one may include a particular leakage post-processing $\mathcal{E}$. This post-treatment may be used to select some particular points in the leakage traces and, possibly, to combine them. For instance, in a second-order advanced SCA involving the mutual information as distinguisher, the function $\mathcal{E}$ can be defined such that $\mathcal{E}\big(\boldsymbol{L}\big) = \big(\boldsymbol{L}[p], \boldsymbol{L}[q]\big)$ for some constant indices (a.k.a. leakage times) p and q. In a second-order advanced SCA involving the correlation coefficient as distinguisher, $\mathcal{E}$ may be defined such that $\mathcal{E}\big(\boldsymbol{L}\big) = (\boldsymbol{L}[p] - \mathbb{E}(\boldsymbol{L}[p])) \cdot (\boldsymbol{L}[q] - \mathbb{E}(\boldsymbol{L}[q]))$. Moreover, the choice of the model function must be done in accordance with the distinguisher (see *e.g.* [52] and [28]).

Example 2. In template A-SCA, the *probability density function* $f_{\hat{\mathbf{s}},x_j}$ of the random variable $(\boldsymbol{L}|\mathbf{S}=\hat{\mathbf{s}}, X=x_j)$ is estimated for all pairs $(\hat{\mathbf{s}}, x_j)$. The model function $\mathbf{m}$ and the predictions $(\boldsymbol{h}_j(\cdot))_j$ are hence defined such that $\boldsymbol{h}_j(\cdot) = \mathbf{m}(o(\hat{\mathbf{s}}, x_j)) = f_{\hat{\mathbf{s}},x_j}(\cdot)$. The estimation may for instance be done on a copy of the attacked device on which an open access is allowed. Eventually, the distinguisher corresponds to a *Maximum Likelihood Test*: $\Delta[\hat{\mathbf{s}}] = \prod_j f_{\hat{\mathbf{s}},x_j}(\boldsymbol{l}_j)$.

2.2 Leakage Measurements and Observations

In the literature, two main approaches have been defined to get the observations $\boldsymbol{l}_j$ (which corresponds to the first step of the attacks in Tables 1 and 2). The first method simply consists in executing the implementation several times (with the same input in simple SCA or with several ones in advanced SCA) and in defining $\boldsymbol{l}_j$ as the observation related to the j^{th} algorithm execution. Those attacks are called *Vertical*. The second method refers to attacks where a single execution is needed and where each $\boldsymbol{l}_j$ corresponds to the observation of a processing at a different time period during the latter. In this case, the index j refers to the time period. The underlying assumption is that all the observations rely on the same internal calculus $O(\mathbf{s}, X)$, parametrized by a same secret $\mathbf{s}$ and different known values x_j in advanced SCA, or a constant one x in simple SCA. Attacks corresponding to this *modus operandi* are called *Horizontal*. Figure 2 illustrates the notations and the differences between the two *modus operandi*.

All the attacks discussed in Section 2.1 can be either Vertical or Horizontal[4]. Even if the Horizontal or Vertical characteristic of an SCA has no impact on the attack steps themselves (as described in Tables 1 and 2), it does impact the implementation security analysis. Indeed, we will see in Section 5 that a countermeasure may become ineffective when going from one category of attacks to another one. We illustrate this in the context of secure RSA implementations.

2.3 Security Evaluation

To analyse the resistance of an implementation against any of the attacks presented in previous sections, we can evaluate the following quantity:

$$\left| \Pr[\mathbf{S}=\mathbf{s}] - \Pr\left[\mathbf{S}=\mathbf{s} \,|\, (\Delta[\hat{\mathbf{s}}])_{\hat{\mathbf{s}}}\right] \right| \ .$$

Indeed, if this value turns out to be negligible, it means that the adversary does not gain any advantage in recovering the secret key $\mathbf{s}$ by processing the SCA than just guessing it at random from uniformly distributed values among the set of all possibilities. In this case, we consider the implementation to be resistant to the corresponding SCA attack (represented by the scores vector $\Delta[\cdot]$). This way of analyzing the security is very close to the approach based on *guessing*

[4] Possibly, the observations acquisition phase may mix horizontal and vertical techniques. In this case, the attack will be termed *Rectangle*.

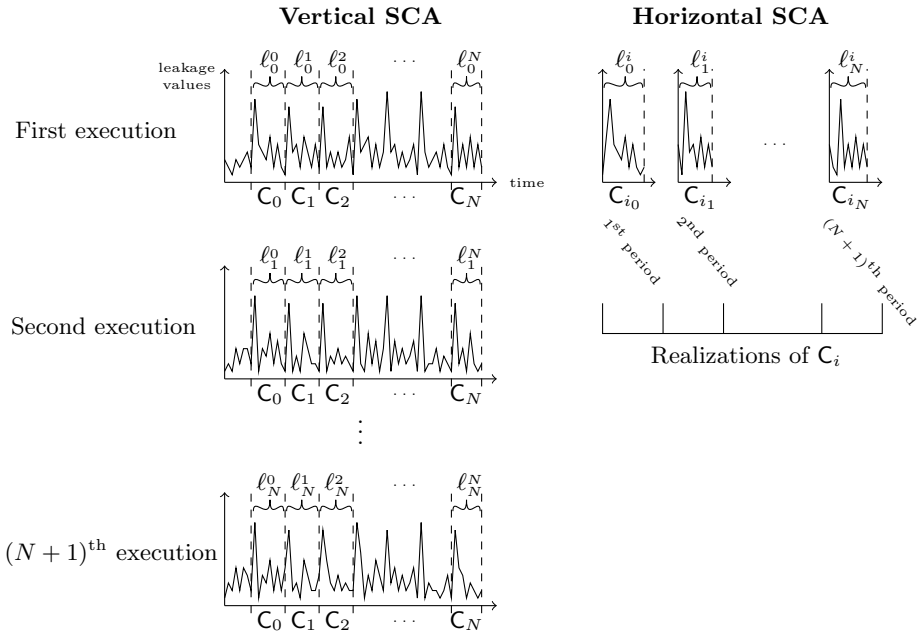

Fig. 2. Vertical and Horizontal SCA

entropy introduced in [55]. Also, after considering $\Delta[\cdot]$ as an *Oracle* with which the adversary interacts, the approach exactly corresponds to that followed in classical security proofs in modern cryptography.

2.4 Taxonomy

Based on the discussions conducted in previous sections, we propose here a general taxonomy for simple and advanced side-channel attacks. To name an attack we propose to use the convention [XXX]-[YYY]-[ZZZ] where:

- XXX equals either S for simple SCA or is a reference to the statistical tool for advanced SCA (*e.g.* C for Correlation, MI for Mutual Information, ML for Maximum Likelihood, LR for Linear Regression, etc.). In case of multivariate SCA, we propose to pad the order/dimension followed by O at the left of the distinguisher letter.
- YYY is an acronym referring to the leakage type; PA for Power Analysis, EMA for Electromagnetic Analysis, TA for Timing Attacks, etc.
- ZZZ is optional and may be used to specify if the attack is profiled or not. In this case, ZZZ is replaced by P (for Profiling) or UnP (for UnProfiling). For instance, Template attack is a ML-PA-P attack.

Of course, all those attacks can be applied either on a Vertical or Horizontal mode. Figure 3 illustrates the taxonomy for some existing attacks.

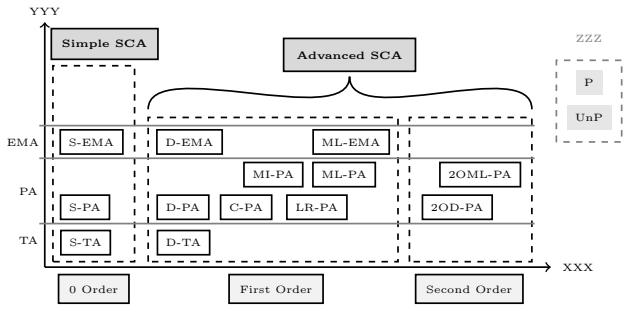

Fig. 3. Side-Channel Attacks

3 Side-Channel Attacks against Elliptic Curves

In this section, we study the resistance of several implementations of `ECC` algorithms with respect to horizontal SCA.

3.1 Background on Elliptic Curves

As this paper focuses on side-channel attacks on `ECC`, let us recall now some basics on elliptic curves and especially on the various ways of representing points on such objects (the reader could refer to [23, 32] for more details).

Throughout this paper, we are interested in elliptic curve implementations running on platforms (ASIC, FPGA, micro-controller) embedding a hardware modular multiplier (*e.g.* a 16-bit, 32-bit or 64-bit multiplier). On such implementations, the considered elliptic curves are usually defined over a prime finite field $\mathbb{F}_p$. In the rest of this paper, we will assume that all curves are defined over $\mathbb{F}_p$ with $p \neq \{2, 3\}$. The algorithm used for the hardware modular multiplication is assumed to be known to the attacker. Moreover, to simplify the attack descriptions, we assume hereafter that the latter multiplication is performed in a very simple way: a schoolbook long integer multiplication followed by a reduction. Most of current devices do not implement the modular multiplications that way, but the attacks described hereafter can always be adapted by changing the definition of the elementary operations of Section 4.3 (see Appendix B).

Definition. An elliptic curve E over a prime finite field $\mathbb{F}_p$ with $p \neq \{2, 3\}$ can be defined as an algebraic curve of affine reduced Weierstrass equation:

$$(E) : y^2 = x^3 + ax + b \ , \tag{3}$$

with $(a, b) \in (\mathbb{F}_p)^2$ and $4a^3 + 27b^2 \neq 0$. Let $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ be two points on (E), the sum $R = (x_3, y_3)$ of P and Q belongs to the curve under a well-known addition rule [37]. The set of pairs $(x, y) \in (\mathbb{F}_p)^2$ belonging to (E), taken with an extra point $\mathcal{O}$, called *point at infinity*, form an abelian group named $E(\mathbb{F}_p)$.

In the rest of the paper, the points will be represented using their projective coordinates. Namely, a point $P = (x, y)$ is expressed as a triplet $(X : Y : Z)$ such that $X = xZ$ and $Y = yZ$. This choice is discussed in Appendix C.

3.2 Points Operations in Presence of SCA

This paper focusses on elliptic curves cryptosystems which involve the scalar multiplication $[\boldsymbol{s}]P$, implemented with the well-known *double and add* algorithm.

In a non-protected implementation, the sequence of point doublings and point additions can reveal the value of $\boldsymbol{s}$ with a single leakage trace. Thus to protect the scheme against `S-PA`, the sequence of point operations must be independent from the secret value. This can be achieved in several ways. The double and add *always* algorithm [25] is the simplest solution. It consists in inserting dummy point additions each time the considered bit value of $\boldsymbol{s}$ is equal to 0. In average, this solution adds an overhead of $\frac{\log_2(\boldsymbol{s})}{2}$ point additions. Another technique consists in using unified formulae for both addition and doubling [10, 11, 43]. Finally, the strategy which is usually adopted in constrained devices such as smart cards is *atomicity* since it achieves the best time/memory trade-off [17, 29, 44]. This principle is a refinement of the double and add always technique. It consists in writing addition and doubling operations as a sequence of a unique pattern. This pattern is itself a sequence of operations over $\mathbb{F}_p$. Since the pattern is unique, the same sequence of field operations is repeated for the addition and the doubling, the only difference being the number of times the pattern is applied for each operation. It thus becomes impossible to distinguish one operation from the other or even to identify the starting and ending of these operations.

To defeat an atomic implementation, the adversary needs to use advanced side-channel attacks (see Section 2.1), such as `D-PA`, `C-PA` and so on. These attacks focus on the operations operands instead of only focusing on the kind of operations. They usually require more observations than for `S-PA` since they rely on statistical analyses. In the `ECC` literature, such attacks have only been investigated in the vertical setting, where they can be efficiently prevented by input randomization.

4 Horizontal Collision Correlation Attack on `ECC`

We show hereafter that implementations combining atomicity and randomization techniques are in fact vulnerable to collision attacks in the horizontal setting. This raises the need for new dedicated countermeasures.

This section starts by recalling some basics on collision attacks. Then, assuming that the adversary is able to distinguish when two field multiplications have a common (possibly unknown) operand, we show how to exhibit flaws in the atomic algorithms proposed in [17, 29, 44] and also in implementations using the unified formulae for Edward's curves [9]. Eventually, we apply the collision attack presented in the first subsection to show how to efficiently deal with the previous assumption.

4.1 Collision Power Analysis in the Horizontal Setting

To recover information on a subpart s of the secret $\boldsymbol{s}$, collision side-channel analyses are usually performed on a sample of observations related to the processing, by the device, of two variables O_1 and O_2 that jointly depend on s. The advantage of those attacks, compared to the classical ones, is that the algorithm inputs can be unknown since the adversary does not need to compute predictions on the manipulated data. When performed in the horizontal setting, the observations on O_1 and O_2 are extracted from the same algorithm execution (see Section 2.1). Then, the correlation between the two samples of observations is estimated thanks to the Pearson's coefficient (see Equation (1)) in order to recover information on s. We sum up hereafter the outlines of this attack, that will be applied in the following.

1. Identify two elementary calculations $\mathsf{C}_1(\cdot)$ and $\mathsf{C}_2(\cdot)$ which are processed several times, say N, with input(s) drawn from the same distribution(s). The correlation between the random variables O_1 and O_2 corresponding to the outputs of C_1 and C_2 must depend on the same secret sub-part s.
2. For each of the N processings of C_1 (resp. C_2) get an observation ℓ_j^1 (resp. ℓ_j^2) with $j \in [1; N]$.
3. Compute the quantity:
$$\hat{\rho} = \hat{\rho}\Big((\ell_j^1)_j, (\ell_j^2)_j\Big)$$
4. Deduce information on s from $\hat{\rho}$.

Table 3. Collision Power Analysis

Remark 3. In Table 3, we use Pearson's coefficient to compare the two samples of observations but other choices are possible (*e.g.* mutual information).

Remark 4. In order to deduce information on s from the knowledge of $\hat{\rho}$, one may use for instance a *Maximum Likelihood* distinguisher (see a discussion on that point in Section 5).

In the next section, the attack in Table 3 is invoked as an Oracle enabling to detect whether two field multiplications share a common operand.

Assumption 1 *The adversary can detect when two field multiplications have at least one operand in common.*

In Section 4.3, we will come back to the latter hypothesis and will detail how it can indeed be satisfied in the particular context of `ECC` implementations on constrained systems.

4.2 Attacks on ECC Implementations: Core Idea.

We start by presenting the principle of the attack on atomic implementations, and then on an implementation based on unified (addition and doubling) formulae over Edward's curves.

Attack on Chevallier-Mames* et al.*'s Scheme. In Chevallier-Mames *et al.*'s atomic scheme, historically the first one, the authors propose the three first patterns[5] given in Figure 4 for the doubling of a point $Q = (X_1 : Y_1 : Z_1)$ and the addition of Q with a second point $P = (X_2 : Y_2 : Z_2)$.

Doubling	Addition
$R_0 \leftarrow a,\ R_1 \leftarrow X_1,\ R_2 \leftarrow Y_1,\ R_3 \leftarrow Z_1$	$R_1 \leftarrow X_1,\ R_2 \leftarrow Y_1,\ R_3 \leftarrow Z_1,$ $R_7 \leftarrow X_2,\ R_8 \leftarrow Y_2,\ R_9 \leftarrow Z_2$
1. $\boldsymbol{R_4 \leftarrow R_1 \cdot R_1}$ $(= X_1 \cdot \boldsymbol{X_1})$	1. $\boldsymbol{R_4 \leftarrow R_9 \cdot R_9}$ $(= Z_2 \cdot \boldsymbol{Z_2})$
$R_5 \leftarrow R_4 + R_4$	$\star$
$\star$	$\star$
$R_4 \leftarrow R_4 + R_5$	$\star$
2. $R_5 \leftarrow R_3 \cdot R_3$	2. $R_1 \leftarrow R_1 \cdot R_4$
$R_1 \leftarrow R_1 + R_1$	$\star$
$\star$	$\star$
$\star$	$\star$
3. $\boldsymbol{R_5 \leftarrow R_5 \cdot R_5}$ $(= Z_1^2 \cdot \boldsymbol{Z_1^2})$	3. $\boldsymbol{R_4 \leftarrow R_4 \cdot R_9}$ $(= Z_2^2 \cdot \boldsymbol{Z_2})$
$\star$	$\star$
$\star$	$\star$
$\star$	$\star$

Fig. 4. Three first atomic patterns of point doubling and addition.

As expected, and as a straightforward implication of the atomicity principle, the doubling and addition schemes perform exactly the same sequence of field operations if the *star* (dummy) operations are well chosen[6]. This implies that it is impossible to distinguish a doubling from an addition by just looking at the sequence of calculations (*i.e.* by `S-PA`). Let us now focus on the operations' operands. In the addition scheme, the field multiplications in Patterns 1 and 3 both involve the coordinate Z_2. On the contrary, the corresponding multiplications in the doubling scheme have *a priori* independent operands (indeed the first one corresponds to the multiplication $X_1 \cdot X_1$, whereas the other one corresponds to $Z_1^2 \cdot Z_1^2$). If an adversary has a mean to detect this difference (which is actually the case under Assumption 1), then he is able to distinguish a doubling from an addition and thus to fully recover the secret scalar $\boldsymbol{s}$. Indeed, let us

[5] For readability reasons we do not recall the full patterns but the interested reader can find them in [17].

[6] Guidelines are given in [17] to define the dummy operations in a pertinent way.

focus on the processing of the second step of the double and add left-to-right algorithm, and let us denote by s the most significant bit of $\boldsymbol{s}$. Depending on s, this sequence either corresponds to the processing of the doubling of $Q = [2]P$ (case $s = 0$) or to the addition of $Q = [2]P$ with P (case $s = 1$). Eventually, the results T_1 and T_2 of the field multiplications in respectively Patterns 1 and 3 satisfy:

$$\begin{cases} T_1 = \left(X_1 \cdot X_1\right)^{1-s} \cdot \left(Z_2 \cdot Z_2\right)^s \\ T_2 = \left(Z_1^2 \cdot Z_1^2\right)^{1-s} \cdot \left(Z_2^2 \cdot Z_2\right)^s \end{cases}, \tag{4}$$

where we recall that we have $P = (X_2 : Y_2 : Z_2)$ and $Q = (X_1 : Y_1 : Z_1)$. Equation (4) and Assumption 1 enables to deduce whether s equals 0 or 1. Applying this attack $\log_2(\boldsymbol{s})$ times, all the bits of $\boldsymbol{s}$ can be recovered one after the other.

We now show that the same idea can successfully be applied to attack the other atomic implementations proposed in the literature, namely those of Longa [44] and Giraud and Verneuil [29].

Attack on Longa's Scheme. The atomic pattern introduced by Longa in [44] is more efficient than that of Chevallier-Mames *et al.* 's scheme. This improvement is got by combining affine and Jacobian coordinates in the points addition, see Figure 5.

It can be seen that the first and third patterns of Longa's scheme contain two field multiplications that either have no operand in common (doubling case) or share the operand Z_1 (addition case). Similarly to Chevallier-Mames *et al.* 's scheme, we can hence define the two following random variables:

$$\begin{cases} T_1 = \left(Z_1 \cdot Z_1\right)^{1-s} \cdot \left(Z_1 \cdot Z_1\right)^s \\ T_2 = \left(X_1 \cdot 4Y_1^2\right)^{1-s} \cdot \left(Z_1^2 \cdot Z_1\right)^s \end{cases}, \tag{5}$$

Under Assumption 1, it leads to the recovery of s.

Attack on Giraud and Verneuil's Scheme. Giraud and Verneuil introduced in [29] a new atomic pattern which reduces the number of field additions, negations and dummy operations ($\star$) compared to the above proposals. The patterns are recalled in Figure 6.

Once again, depending on the secret s, we observe a repetition of two multiplications with a common operand in the first pattern of the addition scheme (ADD 1.), leading to the following equations:

$$\begin{cases} T_1 = \left(X_1 \cdot X_1\right)^{1-s} \cdot \left(Z_2 \cdot Z_2\right)^s \\ T_2 = \left(2Y_1 \cdot Y_1\right)^{1-s} \cdot \left(Z_2^2 \cdot Z_2\right)^s \end{cases}, \tag{6}$$

which, under Assumption 1, leads to the recovery of s.

Remark 5. A second version of the patterns in Figure 6 has been proposed in [29] which allows to save more field additions and negations without addition of dummy operations. This proposal share the same weakness as the previous ones and our attack still applies.

DOUBLING

$R_1 \leftarrow X_1,\ R_2 \leftarrow Y_1,\ R_3 \leftarrow Z_1$

1.
$$\begin{bmatrix} \boldsymbol{R_3 \leftarrow R_3^2} \quad (= Z_1 \cdot \boldsymbol{Z_1}) \\ \star \\ R_5 \leftarrow R_1 + R_4 \\ R_6 \leftarrow R_2^2 \\ R_4 \leftarrow -R_4 \\ R_2 \leftarrow R_2 + R_2 \\ R_4 \leftarrow R_1 + R_4 \end{bmatrix}$$

3.
$$\begin{bmatrix} R_5 \leftarrow R_4^2 \\ \star \\ R_6 \leftarrow R_2 + R_2 \\ \boldsymbol{R_6 \leftarrow R_1 \cdot R_6} \quad (= X_1 \cdot \boldsymbol{4Y_1^2}) \\ R_1 \leftarrow -R_6 \\ R_1 \leftarrow R_1 + R_1 \\ R_1 \leftarrow R_1 + R_5 \end{bmatrix}$$

ADDITION
(mixed coordinates)

Input: $P = (X_1 : Y_1 : Z_1)$ and $Q = (X_2, Y_2)$
Output: $P + Q = (X_3 : Y_3 : Z_3 : X_1' : Y_1')$
$R_1 \leftarrow X_1,\ R_2 \leftarrow Y_1,\ R_3 \leftarrow Z_1,\ R_x \leftarrow X_2,\ R_y \leftarrow Y_2$

1.
$$\begin{bmatrix} \boldsymbol{R_4 \leftarrow R_3^2} \quad (= Z_1 \cdot \boldsymbol{Z_1}) \\ \star \\ \star \\ R_5 \leftarrow R_x \cdot R_4 \\ R_6 \leftarrow -R_1 \\ R_5 \leftarrow R_5 + R_6 \\ \star \end{bmatrix}$$

3.
$$\begin{bmatrix} R_9 \leftarrow R_5 \cdot R_6 \\ \star \\ R_8 \leftarrow R_8 + R_9 \\ \boldsymbol{R_4 \leftarrow R_3 \cdot R_4} \quad (= Z_1^2 \cdot \boldsymbol{Z_1}) \\ \star \\ \star \\ \star \end{bmatrix}$$

Fig. 5. The first and third patterns used in atomicity of Longa

ADDITION

ADD 1.
$$\begin{bmatrix} R_1 \leftarrow Z_2 \cdot \boldsymbol{Z_2} \\ \star \\ \star \\ \star \\ R_2 \leftarrow X_1 \cdot R_1 \\ \star \\ \star \\ \star \\ R_1 \leftarrow R_1 \cdot \boldsymbol{Z_2} \quad (= Z_2^2 \cdot \boldsymbol{Z_2}) \\ \star \\ \star \\ \star \\ \vdots \end{bmatrix}$$

ADD 2.
$$\begin{bmatrix} R_6 \leftarrow R_4^2 \\ \star \\ \star \\ \star \\ R_5 \leftarrow Z_1 \cdot Z_2 \\ \star \\ \star \\ \star \\ Z_3 \leftarrow R_5 \cdot R_4 \\ \star \\ \star \\ \star \\ \vdots \end{bmatrix}$$

DOUBLING

DOUB
$$\begin{bmatrix} R_1 \leftarrow X_1 \cdot \boldsymbol{X_1} \\ R_2 \leftarrow Y_1 + Y_1 \\ \star \\ \star \\ Z_2 \leftarrow R_2 \cdot Z_1 \\ R_4 \leftarrow R_1 + R_1 \\ \star \\ \star \\ R_3 \leftarrow R_2 \cdot Y_1 \quad (= 2Y_1 \cdot \boldsymbol{Y_1}) \\ R_6 \leftarrow R_3 + R_3 \\ \star \\ \star \\ \vdots \end{bmatrix}$$

Fig. 6. The beginning of Giraud and Verneuil's patterns

Attack on Edward's Curves. Edward's representation of elliptic curves has been introduced in [26]. In a subsequent paper [10], Bernstein and Lange homog-

enized the curve equation in order to avoid field inversions in Edward's addition and doubling formulae. For this homogenized representation, points addition and doubling are both computed thanks to the same formula. Let $P = (X_1 : Y_1 : Z_1)$ and $Q = (X_2 : Y_2 : Z_2)$ be two points on the curve, the sum $R = (X_3 : Y_3 : Z_3)$ of P and Q is given by the following system:

$$\begin{cases} X_3 = Z_1 Z_2 (X_1 Y_2 - Y_1 X_2)(X_1 Y_1 Z_2^2 + Z_1^2 X_2 Y_2) \\ Y_3 = Z_1 Z_2 (X_1 X_2 + Y_1 Y_2)(X_1 Y_1 Z_2^2 - Z_1^2 X_2 Y_2) \\ Z_3 = d Z_1^2 Z_2^2 (X_1 X_2 + Y_1 Y_2)(X_1 Y_2 - Y_1 X_2) \end{cases}, \tag{7}$$

where d is some constant related to the Edward curve equation. Formulae (7) works whether P equals Q or not, meaning that it applies similarly for addition and doubling. This is one of the main advantage of Edward's representation compared to the other ones (*e.g.* Projectives) where such unified formulae do not exist. For attack illustration purpose, we give in Figure 7 a sequence of operations which could appear when evaluating the formulae in (7) either to compute $P + P$ or $P + Q$.

Addition
1. $R_1 \leftarrow \mathbf{X_1 Z_2}$
2. $R_2 \leftarrow Y_1 Z_2$
3. $R_3 \leftarrow \mathbf{Z_1 X_2}$
4. $R_4 \leftarrow Z_1 Y_2$
$\vdots$

Fig. 7. First steps of algorithm for addition.

Here, we can exploit the fact that the multiplication $X_1 Z_1$ is performed twice if $P = Q$ (*i.e.* when the formula processed a doubling), which is not the case otherwise (see Figure 7). We can hence define the two following random variables:

$$\begin{cases} T_1 = \left(X_1 \cdot Z_1\right)^{1-s} \cdot \left(X_1 \cdot Z_2\right)^{s} \\ T_2 = \left(X_1 \cdot Z_1\right)^{1-s} \cdot \left(X_2 \cdot Z_1\right)^{s} \end{cases}, \tag{8}$$

which, under Assumption 1, leads to the recovery of s.

Remark 6. This technique still applies in the case of other unified formulae (*e.g.* those introduced in [16]). Indeed, the sequence of operations in [16] present the same weaknesses as illustrated in Figure 7. The multiplication $X_1 Z_1$ is performed twice if the current operation is a doubling (see the first and third multiplications in [16, Sec. 3, Fig. 1]).

In [10], Bernstein and Lange propose a sequence of operations to evaluate (7) while minimizing the total number of multiplications and squarings. We recall it in Figure 8.

ADDITION OR DOUBLING

1. $R_1 \leftarrow Z_1 Z_2$
2. $R_2 \leftarrow R_1^2$
3. $R_3 \leftarrow X_1 X_2$
4. $R_4 \leftarrow Y_1 Y_2$
5. $R_5 \leftarrow dR_3 R_4$
6. $R_6 \leftarrow R_1 - R_5$
7. $R_7 \leftarrow R_1 + R_5$
6. $X_3 \leftarrow R_1 R_6((X_1 + Y_1)(X_2 + Y_2) - R_3 - R_4)$
7. $Y_3 \leftarrow R_1 R_7(R_4 - R_3)$
6. $Z_3 \leftarrow R_6 R_7$

Fig. 8. First steps of algorithm for addition.

For $P = Q$ or $P \neq Q$, it may be checked that the sequence in Figure 8 does not contain two terms which share the same operand. This implies that the attack strategy developed against previous atomic schemes does no longer apply here. It is however possible to follow another attack strategy relying on some defect of the double-and-add algorithm relatively to collision attacks. Indeed, with the double-and-add algorithm, every addition operation leads to perform an addition by a same point : the base point of the scalar multiplication. If we assume that the position in the operations flow of such an addition is known, for example if the most or least significant bit of the exponent is 1, then it is trivial to identify all the positions of the point additions using Assumption 1. Remark that this is quite general : this attack does not rely on some property of the unified sequence of operations described by Bernstein and Lange to compute point additions on Edward curves. It shows that the double-and-add algorithm is vulnerable to the attacks we present even when used with unified formulae.

4.3 Distinguishing Common Operands in Multiplications

In this section we apply the collision attack principle presented in Section 4.1 to show how an adversary may deal with Assumption 1. This will conclude our attack description. As mentioned before, we assume that the field multiplications are implemented in an arithmetic co-processor with a *Long Integer Multiplication* (`LIM`) followed by a reduction. Many other multiplication methods exist but our attack can always be slightly adapted to also efficiently apply to those methods (see Appendix B).

Let ω denote an architecture size (*e.g.* ω equals 8, 16 or 32) and let us denote by $(X[t], \cdots, X[1])_{2^\omega}$ the base-2^ω representation of an integer. We recall hereafter the main steps of the `LIM` when applied between two integers X and Y.

Algorithm 1: Long Integer Multiplication (`LIM`)

Input: $X = (X[t], X[t-1], \ldots, X[1])_{2^\omega}$, $Y = (Y[t], Y[t-1], \ldots, Y[1])_{2^\omega}$.
Output: $\texttt{LIM}(X, Y)$.
for a from 1 to $2t$ **do**
  $R[a] \leftarrow 0$
for a from 1 to t **do**
  $C \leftarrow 0$
  for b from 1 to t **do**
    $(U, V)_{2^\omega} \leftarrow X[a] \cdot Y[b]$ // Operation C_1 (resp. C_2)
    $(U, V)_{2^\omega} \leftarrow (U, V)_{2^\omega} + C$
    $(U, V)_{2^\omega} \leftarrow (U, V)_{2^\omega} + R[a + b - 1]$
    $R[a + b - 1] \leftarrow V$
    $C \leftarrow U$
  $R[a + t] \leftarrow C$
return R

Let W, X, Y and Z be four independent values of size $t\omega$ bits. We show hereafter how to distinguish by side-channel analysis the following three cases:

- Case (0) where the device processes $\texttt{LIM}(X, W)$ and $\texttt{LIM}(Y, Z)$ (all the operands are independent),
- Case (1) where $\texttt{LIM}(X, Z)$ and $\texttt{LIM}(Y, Z)$ are processed (the two `LIM` processings share an operand).
- Case (2) where $\texttt{LIM}(X, Z)$ is processed two times (the two `LIM` processings operate on the same inputs).

To summarize, case (i) corresponds to two integer multiplications sharing i input factor(s).

For such a purpose, and by analogy with our side-channel model in Section 2.1 and Table 3, we denote by C_1 (resp. C_2) the multiplication in the loop during the first `LIM` processing (resp. the second `LIM` processing) and by O_1 (resp. O_2) its result. The output of each ω-bit word multiplication during the loop may be viewed as a realization of the random variable $O^1_{a,b}$ (resp. $O^2_{a,b}$). To each of those realizations we associate a leakage $\ell^1_{a,b}$ (resp. $\ell^2_{a,b}$). To distinguish between cases (1), (2), and (3), we directly apply the attack described in Table 3 and we compute the Pearson's correlation coefficient:

$$\rho\Big((\ell^1_{a,b})_{a,b}, (\ell^2_{a,b})_{a,b}\Big) \ . \tag{9}$$

In the specific case of the `LIM`, where all the word products $X[a]Y[b]$ are processed, on can also average the observations of computations sharing a same input word $Y[b]$. This post-processing on the observations leads to evaluate the following Pearson coefficient in the attack:

$$\rho^{\text{mean}}\Big((\ell^1_{a,b})_{a,b}, (\ell^2_{a,b})_{a,b}\Big) = \rho\left(\Big(\frac{1}{t}\sum_a \ell^1_{a,b}\Big)_b, \Big(\frac{1}{t}\sum_a \ell^2_{a,b}\Big)_b\right) \ . \tag{10}$$

In the following section, we actually argue that this second correlation coefficient gives better results, which is confirmed by our attacks simulations reported in Section 5.

4.4 Study of the Attack Soundness

This section aims at arguing on the soundness of the approach described previously to distinguish common operands in multiplications. For such a purpose, we explicit formulae for the linear correlation coefficients corresponding to Pearson's coefficients given in (9) and (10). Indeed, Pearson's coefficient can be viewed as an estimator of the linear correlation coefficient: when the number of samples tends toward infinity, it tends toward the linear correlation coefficient.

For simplicity, the development is made under the assumption that the device leaks the Hamming weight of the processed data but similar developments could be done for other models and would lead to other expressions. Under the Hamming weight assumption, we have $\ell^1_{a,b} \hookleftarrow \mathrm{HW}(O^1_{a,b}) + B^1_{a,b}$ and $\ell^2_{a,b} \hookleftarrow \mathrm{HW}(O^2_{a,b}) + B^2_{a,b}$ where the $B^i_{a,b}$ random variables are independent Gaussian random variables with zero mean and same standard deviation σ. The three cases presented in last section, 0-, 1- or 2-shared factors in two t-word integers multiplications, can be expressed in the following manner :

- $O^1_{a,b} = X[a].W[b], O^2_{a,b} = Y[a].Z[b]$
- $O^1_{a,b} = X[a].Z[b], O^2_{a,b} = Y[a].Z[b]$
- $O^1_{a,b} = X[a].Z[b], O^2_{a,b} = X[a].Z[b]$

We model the integers X, Y, Z, W as vectors of independent, uniformly distributed, ω-bit word random variables. Furthermore, as seen in the previous subsection, we can either apply the Pearson coefficient directly using (9), or take advantage of the properties of the LIM to aggregate all observations pertaining to a same word of the second factor of the multiplication using (10). In the former case, the pairs $(\ell^1_{a,b}, \ell^2_{a,b})$ can be seen as t^2 samples of a pair of noisy random variables following distributions which depend on the number of shared factors between O^1 and O^2. In the latter case, the pairs of aggregated values can be seen as t samples of pairs of noisy averaged random variables of the form $\frac{1}{t}\sum_{a=1}^{t} \mathrm{HW}(U[a]V[b])$, with distributions depending on the number of shared factors.

For everyone of the 6 cases (3 attack cases times 2 types of correlation processing – normal or with averaging –), the correlation between the two noisy random variables can be expressed as a function of statistical parameters of the Hamming weights of word products (variance and covariance), and the standard deviation σ of the noise random variables. To get the expressions, the bilinearity of the covariance, the independence of the noise random variables and the word random variables may be used, which eventually results in the following formulae:

$$\rho_0 = \rho_0^{\text{mean}} = 0;$$
$$\rho_1 = \frac{\text{cov}(\text{HW}(X.Z), \text{HW}(Y.Z))}{\text{var}(\text{HW}(X.Z)) + \sigma^2}; \qquad \rho_2 = \frac{\text{var}(\text{HW}(X.Z))}{\text{var}(\text{HW}(X.Z)) + \sigma^2};$$
$$\rho_1^{\text{mean}} = \frac{t^2\text{cov}(\text{HW}(X.Z), \text{HW}(Y.Z))}{t\text{var}(\text{HW}(X.Z)) + t(t-1)\text{cov}(\text{HW}(X.Z), \text{HW}(Y.Z)) + t\sigma^2};$$
$$\rho_2^{\text{mean}} = \frac{t\text{var}(\text{HW}(X.Z)) + t(t-1)\text{cov}(\text{HW}(X.Z), \text{HW}(Y.Z))}{t\text{var}(\text{HW}(X.Z)) + t(t-1)\text{cov}(\text{HW}(X.Z), \text{HW}(Y.Z)) + t\sigma^2},$$

where the index of the correlation coefficient refers to the attack case (a.k.a. the number of shared operands between the two word products).

Establishing explicit expressions of the variance and covariance of the Hamming weight of products of uniformly distributed random variables in the general case remains an open question. There are two favourable cases where we are able to obtain such expressions.

The parameter ω is small. The distribution considered can be computed and thus their variance and covariance can be derived. For example for $\omega = 8$, we have

$$\text{var}(\text{HW}(X.Z)) = \frac{1136522959}{2^{28}}$$
$$\text{cov}(\text{HW}(X.Z), \text{HW}(Y.Z)) = \frac{279558159}{2^{28}}$$

Hamming weight of least significant bits. The elementary word multiplication takes as input two ω-bit words and outputs a 2ω-bit word. This output is stored in 2 words. If the leakage associated to the multiplication can be decomposed in two parts, associated to the most significant bits and least significant bits of the results, then word multiplication $\bmod 2^\omega$ can be considered instead of word multiplication in the ring of the integers. Due to the ring structure of $\mathbb{Z}/2^\omega\mathbb{Z}$, explicit formulae giving the variances and covariances of interest as function of ω can be derived.

We have

$$\text{var}(\text{HW}(X.Z)) = \tfrac{1}{2^{2\omega+2}}\left((\omega+1)2^{2\omega} - 2\omega 2^\omega - 1\right),$$
$$\text{cov}(\text{HW}(X.Z), \text{HW}(Y.Z)) = \tfrac{1}{2^{2\omega+2}}\left(2.2^{2\omega} - (2\omega+1)2^\omega - 1\right),$$

$$\rho_1 = \frac{1}{1 + \frac{2^{2\omega+2}\sigma^2 + (\omega-1)2^{2\omega} + 2^\omega}{2.2^{2\omega} - (2\omega+1)2^\omega - 1}}, \quad \rho_1^{\text{mean}} = \frac{1}{1 + \frac{1}{t}\frac{2^{2\omega+2}\sigma^2 + (\omega-1)2^{2\omega} + 2^\omega}{2.2^{2\omega} - (2\omega+1)2^\omega - 1}},$$
$$\rho_2 = \frac{1}{1 + \frac{2^{2\omega+2}\sigma^2}{(\omega+1)2^{2\omega} - 2\omega 2^\omega - 1}}, \qquad \rho_2^{\text{mean}} = \frac{1}{1 + \frac{2^{2\omega+2}\sigma^2}{t(2.2^{2\omega} - (2\omega+1)2^\omega - 1) + 2^\omega[(\omega-1)2^\omega + 1]}}.$$

Note that when t tends towards infinity, the correlation coefficient of averaged variables tends towards 1 (which is optimal), whereas the correlation coefficient when considering directly the random variables has some value strictly lower than 1 independently of the size of the sample.

5 Experiments

In order to validate the approach presented in Section 4.3 and thus to illustrate the practical feasibility of our attack, we performed several simulation campaigns for various sizes of elliptic curves, namely $\lceil \log_2(p) \rceil \in \{160, 256, 384\}$, implemented on different kinds of architectures, namely $\omega \in \{8, 32\}$ using the Chevallier-Mames *et al.* 's scheme. Each experiment has been performed in the same way. For each (p, ω), we computed Pearson's correlation coefficients (9) and (10) between the sample of observations coming from the leakages on operations C_1 and C_2 in the two following cases:

- when the secret bit s is equal to 1, that is when an addition is performed (which implies correlated random variables, see (4)),
- when the secret bit s is equal to 0, that is when a doubling operation is performed (which implies independent random variables, see (4)).

From the configuration (p, ω), the size t of the observations' samples used in the attack can be directly deduced: it equals $\lceil \frac{\log_2(p)}{\omega} \rceil$. The quality of the estimations of the correlation coefficient by Pearson's coefficient depends on both the observations *signal to noise ratio* (SNR) and t. When the SNR tends towards 0, the sample size t must tend towards infinity to deal with the noise. Since, in our attack the samples size cannot be increased (it indeed only depends on the implementation parameters p and ω), our correlation estimations tend towards zero when the SNR decreases. As a consequence, distinguishing the two Pearson coefficients coming from $s = 0$ and $s = 1$ becomes harder when the SNR decreases. This observation raises the need for a powerful (and robust to noise) test to distinguish the two coefficients. To take this into account for each setting (p, ω) and several SNR, we computed an histogram approximation of the distribution of Pearson's coefficients defined in (9) and (10) over samples of size t. To build those kinds of templates, leakages have been generated in the Hamming weight model with additive Gaussian noise of mean 0 and standard deviation[7] σ. When there is no noise at all, namely when $\sigma = 0$ (*i.e.* $\mathtt{SNR} = +\infty$), one can observe that the mean of Pearson's coefficient is coherent with the predictions evaluated in Section 4.4.

Figures (9-12) illustrate the spreading of the obtained Pearson's coefficients. The curves indicate the evolution of the maxima of the distributions, and the colored cone around the maximum indicates the smallest interval containing more than half of the probability weight of the Pearson's coefficient distribution. This gives us information about the amount of trust we can put into the values obtained during the attacks. It also shows whether a distinction between the right hypothesis and the wrong one can easily be highlighted. For each SNR value (denoted by τ) and each sample size t, let us denote by $\hat{\rho}_{0,t}(\tau)$ (resp. $\hat{\rho}_{1,t}(\tau)$) the random variable associated to the processing of (9) for $s = 0$ (resp. for $s = 1$). Clearly, the efficiency of the attack described in Section 4 depends on the ability of the adversary to distinguish, for a fixed pair (t, τ), the distribution

[7] In this context, the SNR simply equals $\omega/4\sigma^2$.

of $\hat{\rho}_{0,t}(\tau)$ from that of $\hat{\rho}_{1,t}(\tau)$. In other terms, once the adversary has computed a Pearson coefficient $\hat{\rho}$ he must decide between the two following hypotheses; $\mathrm{H}_0 : \hat{\rho} \hookleftarrow \hat{\rho}_{0,t}(\tau)$ or $\mathrm{H}_1 : \hat{\rho} \hookleftarrow \hat{\rho}_{1,t}(\tau)$. For such a purpose, we propose here to apply a *maximum likelihood* strategy and to choose the hypothesis having the highest probability to occur. Based on the approximation of the Pearson's coefficient we obtained, we computed the value $\rho_t^{\mathrm{limit}}(\tau)$ for which the values of the density probability function in both hypotheses are equal. During the attack, if $\hat{\rho}$ is smaller than $\rho_t^{\mathrm{limit}}(\tau)$, the distinguisher chooses H_0, otherwise it chooses H_1. Attacks reported in Figures 13 and 14 apply this strategy. They aim at recovering one bit of the secret scalar.

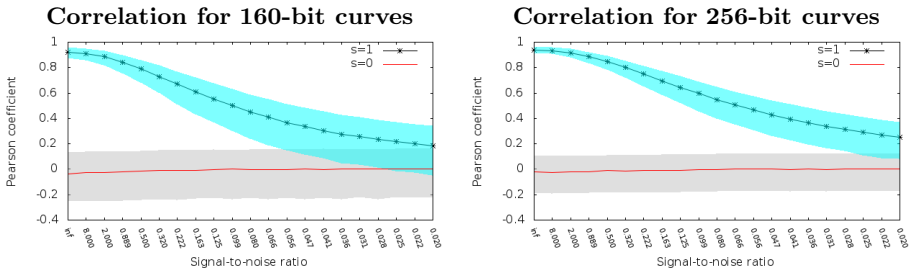

Fig. 9. Pre-computations on w =8-bit registers

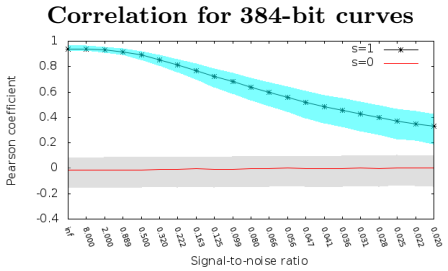

Fig. 10. Pre-computations on w =8-bit registers

Remark 7. Since the adversary is not assumed to know the exact leakage `SNR`, the maximum likelihood can be computed for several `SNR` values τ starting from ∞ to some pre-defined threshold. This problematic occurs each time that the principle of collision attacks is applied.

Remark 8. For a curve of size $n = \lceil \log_2(p) \rceil$ and a ω-bit architecture, the adversary can have a sample of $t = \lceil \frac{n}{\omega} \rceil$ observations if he averages over the columns and $t = \lceil (\frac{n}{\omega})^2 \rceil$ without averaging. All experiments provided in this section have been performed using the "average" strategy.

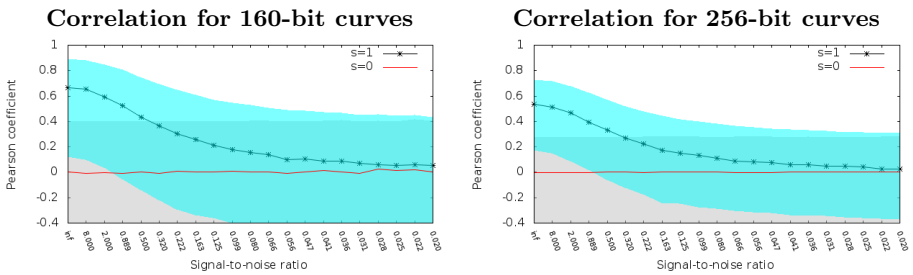

Fig. 11. Pre-computations on w =32-bit registers

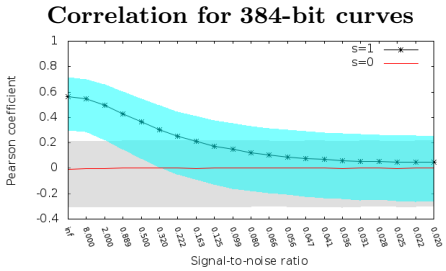

Fig. 12. Pre-computations on w =32-bit registers

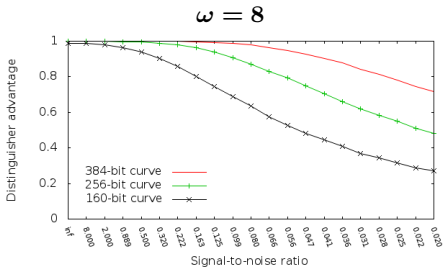

Fig. 13. Success rate of the attack on 8-bit registers

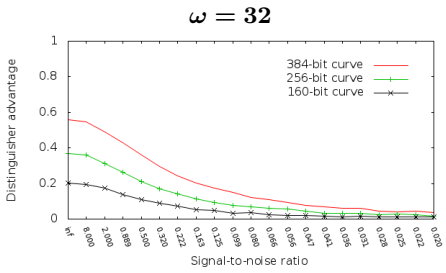

Fig. 14. Success rate of the attack on 32-bit registers

This attack works for any kind of architecture, even for a 32-bit one (see Fig. 14), which is the most common case in nowadays implementations. In the presence of noise, the attack success decreases highly but stays quite successful for curves of size 160, 256 and 384 bits. In all experiments (Fig. 13-14), we also observe that the success rate of our attack increases when the size of the curve becomes larger. This behaviour can be explained by the increasing number of observations available in this case. Paradoxically, it means that when the theoretical level of security becomes stronger (*i.e.* p is large), resistance against side-channel attacks becomes weaker. This fact stands in general for horizontal attacks and has already been noticed in [19, 58].

6 Discussion About Possible Countermeasures

In this section we first recall classical countermeasures that are usually involved to defeat simple SCA and vertical advanced SCA, and we discuss about their (in)efficiency in the horizontal setting. In particular, following the same reasoning as in [20], we alert on the fact that a countermeasure effectiveness can be annihilated when going from the vertical context to the horizontal one. Then, in Section 6.2, we particularly focus on several countermeasures dedicated to Horizontal advanced SCA, trying to identify those that are the most effective against the collisions attack proposed in this paper.

6.1 Overview of Classical Countermeasures on Elliptic Curves

Careful choice of the elliptic curve scalar multiplication. A first natural idea is to look for a scalar multiplication scheme inherently resistant against our horizontal collisions attack. The choice of a regular scheme (always performing the same sequence of additions and doublings whatever the secret scalar) seems *a priori* pertinent as distinguishing the two operations brings no sensitive information. From this point of view, the schemes *Double & Add Always* [25] or the *Montgomery Ladder* [49] look interesting. We recall them hereafter:

Algorithm 2: Double & Add Always Scalar Multiplication

Input: the public point P and a secret scalar $\mathbf{s} = (s_0, s_1, \cdots, s_{\ell-1})_2$.
Output: $\mathbf{s}P$.

$R_0 \leftarrow \mathcal{O}$
for $i = \ell - 1$ **to** 0 **do**
  $R_0 \leftarrow 2R_0$
  $R_{1-s_i} \leftarrow R_0 + P$
return R_0

Algorithm 3: Montgomery Ladder

Input: the public point P and a secret scalar $\mathbf{s} = (s_0, s_1, \cdots, s_{\ell-1})_2$.
Output: $\mathbf{s}P$.

$R_0 \leftarrow \mathcal{O}$
$R_1 \leftarrow P$
for $i = \ell - 1$ **to** 0 **do**
  $R_{1-s_i} \leftarrow R_0 + R_1$
  $R_{s_i} \leftarrow 2R_{s_i}$
return R_0

Unfortunately, even if the above algorithms withstand straightforward applications of our attack, they stay vulnerable to a slight adaptation of it. Let us respectively denote by $R_0^{(i)}$ and $R_1^{(i)}$ the values of the registers R_0 and R_1 after the i^{th} iteration of the loop.

- For Algorithm 2, we have $R_0^{(i+1)} = 2R_0^{(i)}$. This doubling and the addition performed to compute $R_1^{(i)}$ have the first operand $R_0^{(i)}$ in common iff $s_i = 0$ (otherwise one can assume that they operate on independent operands). It is therefore possible to recover the value of each bit s_i by applying the idea of our horizontal collisions attack[8] to the sequences of field operations involved in both $R_0^{(i+1)} \leftarrow 2R_0^{(i)}$ and $R_1^{(i)} \leftarrow R_0^{(i)} + P$.
- For Algorithm 3, we have $R_{1-s_{i+1}}^{(i+1)} = R_0^{(i)} + R_1^{(i)}$. This addition and the subsequent doubling performed to compute $R_{s_i}^{(i+1)}$ have the operand $R_0^{(i)}$ in common iff $s_{i+1} = 0$. It is therefore possible to recover the value of each bit s_{i+1} by applying the idea of our horizontal collisions attack to the manipulations of the field coordinates of the first operand in both $R_{1-s_{i+1}}^{(i+1)} \leftarrow R_0^{(i)} + R_1^{(i)}$ and $R_{s_{i+1}}^{(i+1)} \leftarrow 2R_{s_{i+1}}^{(i)}$. The same kind of flaw can also be found in the left-to-right version of the Montgomery Ladder proposed in [35].

This attack is defeated if the step $R_{1-s_i} \leftarrow R_0 + R_1$ in Algorithm 3 is replaced by $R_{1-s_i} \leftarrow R_{1-s_i} + R_{s_i}$ (which is actually the way Montgomery Ladder is classically described in the SCA literature – *e.g.* [31] –).

Randomizing the scalar. This countermeasure was proposed by Coron in [25]. It consists in changing the value of the secret scalar in the point multiplication for each computation. For most schemes based on elliptic curves this countermeasure may be viewed as part of the protocol since the secret used in the point multiplication changes at each execution of the protocol. This is for instance the case with `ECDH` and `ECDSA`. This explains why, usually, specific countermeasures

[8] Contrary to the attacks described in Section 4, the attack against Algorithms 2 and 3 does not try to detect two similar operations with a common operand but tries to detect when a same operand is manipulated two times. Even if this scenario is not exactly the one analyzed in this paper, we think that the corresponding attack stays efficient as it is based on the same principles.

against advanced (vertical) attacks are not implemented in the elliptic curve setting. When such a countermeasure is added on purpose or part of the protocol, it does not provide any protection against our attack since it only requires one power curve.

Blinding the base point. This is the second countermeasure proposed by Coron in [25]. It consists in modifying the point P by adding a random point R. This countermeasure does not have any impact against our attack since the adversary recovers the value of the secret exponent independently from the base point value.

Randomizing the coordinates. This third countermeasure of Coron [25], which modifies the coordinates of the point P, has no effect on our attack exactly for the same reasons as for the second countermeasure. Indeed, the secret scalar is recovered independently from the base point value.

Splitting the scalar. It is considered in [18, 22]. It consists in splitting the scalar in two parts *i.e.* in computing $[\boldsymbol{s}]P = [\boldsymbol{s}_1]P + [\boldsymbol{s}_2]P$. This countermeasure decreases the vertical attacks efficiency by a quadratic factor since the two point multiplications need to be combined in a so-called *second-order* attack setting. Against our attack, this countermeasure is much less efficient. Indeed, its efficiency is only decreased by a factor 2 since we are able to recover the bits of $\boldsymbol{s}_1$ first, then those of $\boldsymbol{s}_2$ in an independent way.

Randomizing the curve or the field. This idea is described in [4, 34, 56] with different techniques. All these techniques modify the input of the point multiplication but they do not hide the property exploited in most of our attacks, that is the reuse of the point coordinate Z in the addition operation, and thus turn out to be inefficient in our setting.

6.2 Investigating Countermeasures inside Modular Multiplication

The previous section has raised the need for new techniques to thwart side-channel analysis in the horizontal setting. In this section, we deal with this issue by investigating whether the solutions proposed in [19, 20] and developed in [57, Sec. 2.7] in the context of `RSA` can be applied to `ECC`.

As discussed in Section 4.3 and Appendix B, the multiplication of two integers $U = (U[t], \cdots U[1])_{2^\omega}$ and $V = (V[t], \cdots V[1])_{2^\omega}$ frequently leads to the processing of the following ω-bit word multiplications:

$$\begin{pmatrix} U[1]V[1] & U[1]V[2] & \cdots & U[1]V[t] \\ U[2]V[1] & \cdots & \cdots & U[2]V[t] \\ \vdots & \vdots & \ddots & \vdots \\ U[t]V[1] & \cdots & \cdots & U[t]V[t] \end{pmatrix} .$$

Essentially, our attack consists in detecting when the same value V is used for two different modular multiplications. For such a purpose we correlate the elements of the previous matrix with those of the following one:

$$\begin{pmatrix} U'[1]V'[1] & U'[1]V'[2] & \cdots & U'[1]V'[t] \\ U'[2]V'[1] & \cdots & \cdots & U'[2]V'[t] \\ \vdots & \vdots & \ddots & \vdots \\ U'[t]V'[1] & \cdots & \cdots & U'[t]V'[t] \end{pmatrix},$$

where $V = V'$ for $s = 1$ and $V \neq V'$ for $s = 0$.

Countermeasures proposed in [19, 20, 57] consist in protecting the $U[i]V[j]$ (resp. $(U'[i]V'[j])$) products by blinding some of the operands with different random values and/or by randomizing the order in which they are processed. We study hereafter the soundness of those techniques when applied in `ECC` context.

Operands blinding. This countermeasure blinds each $U[i]$ and $V[j]$ with two random values R_1 and R_2 such that $U[i]V[j] = (U[i]-R_1)(V[j]-R_2)+R_1V[j]+R_2U[i]-R_1R_2$. Then, each of the four terms is computed independently. First of all, it must be noticed that the multiplicative masking of $V[j]$ (by R_1) and $U[i]$ (by R_2) is not effective when $U[i]$ or $V[j]$ is null, which introduces a flaw that may be exploited by `C-PA` [30]. Moreover, the values $\frac{1}{t}(\sum_i U[i])(V[j]-R_2)$ are still correlated with the values $\frac{1}{t}(\sum_i U'[i])(V'[j]-R'_2)$ when $V = V'$. Thus, although this countermeasure decreases the attack efficiency, it does not totally remove the leakage.

Shuffling rows and columns. In this countermeasure, rows and columns of the matrix are permuted independently. This means that one permutation applies on the rows and a second one on the columns. However one can notice that the rows permutation is the same for each column and reciprocally. Shuffling rows has no impact since U is unknown in the attack. When averaging over each column of the matrix, we observe that $\left(\frac{1}{t}(\sum_j U[j])V[i]\right)_i$ and $\left(\frac{1}{t}(\sum_j U'[j])V'[i]\right)_i$ stay correlated when $V[i] = V'[i]$. However, due to the column permutation, the adversary needs to guess the value of the permutation in order to observe this correlation. As a consequence, this countermeasure adds a $t!$ search factor to the computational time of the attack.

Shuffling and blinding. In this countermeasure, only the $V[j]$ are blinded using t independent random values associated with each row, while the rows of the matrix are randomly permuted. Namely, the $U[i]$ are blinded while the columns of the matrix are permuted. This countermeasure prevents our attack, but opens new issues such as the manipulation of the correcting factor related to the blinding part. This value could be exploited in a zero-value attack [30].

Global shuffling. This countermeasure, proposed in [7], starts from the shuffling and blinding countermeasure, but performs the shuffling of rows and columns simultaneously. This essentially replaces two permutations of size t by a single one of size t^2, hence increasing the security against brute force attack from $2(t!)$ to $(t^2!)$. Of course, care should also be taken when propagating the carry during the reconstruction of UV from the $U[i]V[j]$. This point is successfully addressed in [7]. Eventually, among shuffling methods, this countermeasure seems to be the most interesting one to defeat our attack. It is however still an open problem to formally quantify its effectiveness.

Conclusion

In this paper, we investigated the horizontal correlation attacks, introduced by Clavier *et al.* , in the context of `ECC` implementations. We showed that these attacks, although fundamentally belonging to the well-known advanced side-channel attacks, are not covered by traditional countermeasures such as randomization techniques. Indeed, we have shown that we are able to apply such horizontal attacks on state-of-the-art SCA-protected `ECC` implementations. We showed how to defeat all the atomic point addition and doubling schemes proposed in the literature and also the one using the unified formulas, even when combined with randomization. Our simulations confirmed the validity of our attack for classical sizes of curves. It stays applicable even in a moderate noise setting.

We feel that this topic opens many areas for further research. Namely, the formal study and proofs of countermeasures against horizontal attacks is necessary in order to effectively protect implementations against this kind of attacks. It would also be of interest to investigate the applicability of such attacks to other domains of cryptography, such as pairings, code-based cryptography or keyed hash functions.

References

1. Karatsuba A. and Ofman Yu., editors. *Multiplication of Many-Digital Numbers by Automatic Computers*, volume 145, 1962.
2. ANSI X9.62. *Public Key Cryptography for The Financial Service Industry : The Elliptic Curve Digital Signature Algorithm (ECDSA).* American National Standards Institute, 1998.
3. ANSI X9.63. *Public Key Cryptography for The Financial Service Industry : Key Agreement and Key Transport Using Elliptic Curve Cryptography.* American National Standards Institute, 1998.
4. Yoo-Jin Baek and Ihor Vasyltsov. How to Prevent DPA and Fault Attack in a Unified Way for ECC Scalar Multiplication - Ring Extension Method. In *ISPEC*, pages 225–237, 2007.
5. Paul Barrett. Implementing the Rivest Shamir and Adleman public key encryption algorithm on a standard digital signal processor. In *Proceedings on Advances in cryptology—CRYPTO '86*, pages 311–323, London, UK, UK, 1987. Springer-Verlag.

6. L. Batina, B. Gierlichs, E. Prouff, M. Rivain, F.-X. Standaert, and N. Veyrat-Charvillon. Mutual information analysis: a comprehensive study. *to appear in the Journal of Cryptology*, 24(2):269–291, April 2011.
7. Aurélie Bauer, Éliane Jaulmes, Emmanuel Prouff, and Justine Wild. Horizontal and Vertical Side-Channel Attacks against Secure RSA Implementations. In Ed Dawson, editor, *Topics in Cryptology — CT-RSA 2013*, volume 7779 of *Lecture Notes in Computer Science*, pages 1–17. Springer, 2013.
8. Aurélie Bauer, Éliane Jaulmes, Emmanuel Prouff, and Justine Wild. Horizontal collision correlation attack on elliptic curves. In Tanja Lange, Kristin E. Lauter, and Petr Lisonek, editors, *Selected Areas in Cryptography*, volume 8282 of *Lecture Notes in Computer Science*, pages 553–570. Springer, 2013.
9. Daniel J. Bernstein and Tanja Lange. Analysis and optimization of elliptic-curve single-scalar multiplication. Cryptology ePrint Archive, Report 2007/455, 2007. `http://eprint.iacr.org/`.
10. Daniel J. Bernstein and Tanja Lange. Faster Addition and Doubling on Elliptic Curves. In K. Kurosawa, editor, *Advances in Cryptology — Proceedings of ASIACRYPT 2007*, volume 4833 of *Lecture Notes in Computer Science*, pages 29–50. Springer, 2007.
11. O. Billet and M. Joye. The Jacobi Model of an Elliptic Curve and Side-Channel Analysis. Cryptology ePrint Archive, Report 2002/125, 2002.
12. Andrey Bogdanov, Ilya Kizhvatov, and Andrei Pyshkin. Algebraic Methods in Side-Channel Collision Attacks and Practical Collision Detection. In Dipanwita R. Chowdhury, Vincent Rijmen, and Abhijit Das, editors, *Progress in Cryptology – INDOCRYPT 2008*, volume 5365 of *LNCS*, pages 251–265. SV, 2008.
13. A. Booth. A Signed Binary Multiplication Technique. *Quarterly Journal of Mechanics and Applied Mathematics*, 4(2):236–240, June 1951.
14. Ernest F. Brickell. A Survey of Hardware Implementation of RSA (Abstract). In *CRYPTO*, volume 435 of *Lecture Notes in Computer Science*, pages 368–370. Springer, 1989.
15. É. Brier, C. Clavier, and F. Olivier. Correlation Power Analysis with a Leakage Model. In M. Joye and J.-J. Quisquater, editors, *Cryptographic Hardware and Embedded Systems – CHES 2004*, volume 3156 of *Lecture Notes in Computer Science*, pages 16–29. Springer, 2004.
16. É. Brier and M. Joye. Weierstraß Elliptic Curves and Side-Channel 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, 2002.
17. Benoît Chevallier-Mames, Mathieu Ciet, and Marc Joye. Low-cost Solutions for Preventing Simple Side-Channel Analysis: Side-Channel Atomicity. *IEEE Transactions on Computers*, 53(6):760–768, 2004.
18. M. Ciet and M. Joye. Elliptic Curve Cryptosystems in the Presence of Permanent and Transient Faults. Cryptology ePrint Archive, Report 2003/028, 2003.
19. Christophe Clavier, Benoît Feix, Georges Gagnerot, Christophe Giraud, Mylène Roussellet, and Vincent Verneuil. ROSETTA for Single Trace Analysis – Recovery of Secret Exponent by Triangular Trace Analysis. In *INDOCRYPT*, pages 140–155, 2012.
20. Christophe Clavier, Benoit Feix, Georges Gagnerot, Mylène Roussellet, and Vincent Verneuil. Horizontal correlation analysis on exponentiation. In Miguel Soriano, Sihan Qing, and Javier López, editors, *ICICS*, volume 6476 of *Lecture Notes in Computer Science*, pages 46–61. Springer, 2010.

21. Christophe Clavier, Benoît Feix, Georges Gagnerot, Mylène Roussellet, and Vincent Verneuil. Improved Collision-Correlation Power Analysis on First Order Protected AES. In Bart Preneel and Tsuyoshi Takagi, editors, *Cryptographic Hardware and Embedded Systems, 13th International Workshop – CHES 2011*, volume 6917 of *Lecture Notes in Computer Science*, pages 49–62. Springer, 2011.
22. Christophe Clavier and Marc Joye. Universal Exponentiation Algorithm – A First Step towards Provable SPA-Resistance. In Koç et al. [39], pages 300–308.
23. Henri Cohen and Gerhard Frey, editors. *Handbook of Elliptic and Hyperelliptic Curve Cryptography*. CRC Press, 2005.
24. Paul G. Comba. Exponentiation Cryptosystems on the IBM PC. *IBM Systems Journal*, 29(4):526–538, 1990.
25. J.-S. Coron. Resistance against Differential Power Analysis for Elliptic Curve Cryptosystems. In Ç.K. Koç and C. Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES '99*, volume 1717 of *Lecture Notes in Computer Science*, pages 292–302. Springer, 1999.
26. Harold M. Edwards. A normal form for elliptic curves. *Bulletin of the American Mathematical Society*, 44:393–422, 2007.
27. K. Gandolfi, C. Mourtel, and F. Olivier. Electromagnetic Analysis: Concrete Results. In Koç et al. [39], pages 251–261.
28. Benedikt Gierlichs, Lejla Batina, Pim Tuyls, and Bart Preneel. Mutual information analysis. In Elisabeth Oswald and Pankaj Rohatgi, editors, *CHES*, volume 5154 of *Lecture Notes in Computer Science*, pages 426–442. Springer, 2008.
29. Christophe Giraud and Vincent Verneuil. Atomicity Improvement for Elliptic Curve Scalar Multiplication. In D. Gollmann, J.-L. Lanet, and J. Iguchi-Cartigny, editors, *Smart Card Research and Advanced Applications, 9th International Conference – CARDIS 2010*, volume 6035 of *Lecture Notes in Computer Science*, pages 80–101. Springer, 2010.
30. J. Golić and C. Tymen. Multiplicative Masking and Power Analysis of AES. In B.S. Kaliski Jr., Ç.K. Koç, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2002*, volume 2523 of *Lecture Notes in Computer Science*, pages 198–212. Springer, 2002.
31. Raveen R. Goundar, Marc Joye, Atsuko Miyaji, Matthieu Rivain, and Alexandre Venelli. Scalar multiplication on weierstraß elliptic curves from co-z arithmetic. *J. Cryptographic Engineering*, 1(2):161–176, 2011.
32. D. Hankerson, A.J. Menezes, and S. Vanstone. *Guide to Elliptic Curve Cryptography*. Springer Professional Computing Series, January 2003.
33. ISO/IEC JTC1 SC17 WG3/TF5 for the International Civil Aviation Organization. *Supplemental Access Control for Machine Readable Travel Documents. Technical Report*, 2010.
34. M. Joye and C. Tymen. Protections against Differential Analysis for Elliptic Curve Cryptography. In Koç et al. [39], pages 386–400.
35. Marc Joye. Highly regular right-to-left algorithms for scalar multiplication. In Pascal Paillier and Ingrid Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems – CHES 2007*, volume 4727 of *Lecture Notes in Computer Science*, pages 135–147. Springer, 2007.
36. D.E. Knuth. *The Art of Computer Programming*, volume 2. Addison Wesley, third edition, 1988.
37. N. Koblitz. Elliptic Curve Cryptosystems. *Mathematics of Computation*, 48(177):203–209, 1987.
38. C.K. Koc. *Cryptographic Engineering*. Springer, 2008.

39. Ç.K. Koç, D. Naccache, and C. Paar, editors. *Cryptographic Hardware and Embedded Systems – CHES 2001*, volume 2162 of *Lecture Notes in Computer Science*. Springer, 2001.
40. P. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In N. Koblitz, editor, *Advances in Cryptology – CRYPTO '96*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113. Springer, 1996.
41. P. Kocher, J. Jaffe, and B. Jun. Differential Power Analysis. In M.J. Wiener, editor, *Advances in Cryptology – CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999.
42. Paul C. Kocher, Joshua Jaffe, Benjamin Jun, and Pankaj Rohatgi. Introduction to Differential Power Analysis. *J. Cryptographic Engineering*, 1(1):5–27, 1998.
43. P.-Y. Liardet and N.P. Smart. Preventing SPA/DPA in ECC Systems Using the Jacobi Form. In Koç et al. [39], pages 401–411.
44. Patrick Longa. Accelerating the Scalar Multiplication on Elliptic Curve Cryptosystems over Prime Fields. Master's thesis, School of Information Technology and Engineering, University of Ottawa, Canada, 2007.
45. Stefan Mangard and François-Xavier Standaert, editors. *Cryptographic Hardware and Embedded Systems, CHES 2010, 12th International Workshop, Santa Barbara, CA, USA, August 17-20, 2010. Proceedings*, volume 6225 of *Lecture Notes in Computer Science*. Springer, 2010.
46. Silvio Micali and Leonid Reyzin. Physically Observable Cryptography (Extended Abstract). In Moni Naor, editor, *Theory of Cryptography Conference – TCC 2004*, volume 2951 of *Lecture Notes in Computer Science*, pages 278–296. Springer, 2004.
47. V.S. Miller. Use of Elliptic Curves in Cryptography. In H.C. Wiliams, editor, *Advances in Cryptology – CRYPTO '85*, volume 218 of *Lecture Notes in Computer Science*, pages 417–426. Springer, 1985.
48. P.L. Montgomery. Modular multiplication without trial division. *Math. Comp.*, 44(170):519–521, April 1985.
49. P.L. Montgomery. Speeding the Pollard and Elliptic Curve Methods of Factorization. *Mathematics of Computation*, 48:243–264, 1987.
50. Amir Moradi. Statistical tools flavor side-channel collision attacks. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT*, volume 7237 of *Lecture Notes in Computer Science*, pages 428–445. Springer, 2012.
51. Amir Moradi, Oliver Mischke, and Thomas Eisenbarth. Correlation-enhanced power analysis collision attack. In Mangard and Standaert [45], pages 125–139.
52. Emmanuel Prouff, Matthieu Rivain, and Régis Bévan. Statistical Analysis of Second Order Differential Power Analysis. *IEEE Transactions on Computers*, 58(6):799–811, 2009.
53. Jean-Jacques Quisquater and D. Samyde. 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., 2000.
54. Kai Schramm, Thomas Wollinger, and Christof Paar. A New Class of Collision Attacks and its Application to DES. In T. Johansson, editor, *Fast Software Encryption – FSE 2003*, volume 2887 of *Lecture Notes in Computer Science*, pages 206–222. Springer, 2003.
55. Francois-Xavier Standaert, Tal G. Malkin, and Moti Yung. A Unified Framework For The Analysis of Side-Channel Attacks. In *EUROCRYPT*, volume 5479 of *Lecture Notes In computer Science*, pages 443–461. Springer, 2009.
56. Michael Tunstall and Marc Joye. Coordinate Blinding over Large Prime Fields. In Mangard and Standaert [45], pages 443–455.

57. Vincent Verneuil. *Elliptic Curve Cryptography and Security of Embedded Devices*. PhD thesis, Université de Bordeaux, 2012.
58. Colin D. Walter. Sliding Windows Succumbs to Big Mac Attack. In Koç et al. [39], pages 286–299.

A Extension to Higher Orders

The leakage definition given in (2) stands for contexts where instantaneous leakage about the implementation secret parameter exists. When the latter condition is not verified, the adversary must consider several intermediate values simultaneously to reveal sensitive information. In this context, side-channels are usually called *multivariate* by opposition with the first class of attacks that are called *univariate*. Except (2), the framework introduced in Section 2.1 and the formalism given in Section 2.2 continue to be valid. For completeness, we generalize the definition of the leakage coordinates in (2) to encompass contexts where several intermediate results must be observed to reveal information about a sensitive internal processing $O(k, X)$:

$$\boldsymbol{L}[i] = \varphi_i\big(V_i\big) + \beta_i \tag{11}$$

where φ_i and β_i are similar as in (2) and where V_i refers to the value manipulated at time t_i. [9]

When the general definition (11) is used in place of (2) to model the instantaneous leakage, a prerequisite for a multivariate SCA to be possible is that there exists at least one tuple of coordinates of $\boldsymbol{L}$ that statistically depends on $O(\mathbf{s}, X)$. Actually for Horizontal SCA the number of tuples must be high enough for the involved statistical tools to be effective.

B Implementations of Modular Multiplications

In Section 4.3, we argued that an adversary may deal with Assumption 1 by using collisions attack. For such a purpose, we focussed on the classical Long Integer Multiplication (`LIM`) and we showed that horizontal collisions attacks can be applied to distinguish when two multiplications are performed with at least one common operand. Obviously, in practice, there are several other techniques to implement the modular multiplication $U \cdot V \bmod p$ between two $t\omega$-bit long integers. Let us argue here briefly that our attack still applies efficiently in some of these other cases.

Among all existing modular multiplication techniques, two main methods can be highlighted: those which perform long multiplications [1, 13, 24] followed

[9] For instance, if $\boldsymbol{L}$ is related to the manipulation of two shares M_1 and M_2 of O, then one can for instance assume that half of the V_i corresponds to M_1 and the other half to M_2. Moreover, (2) is a particular case of (11) where all manipulated data are assumed to be equal to O.

by a global reduction [5, 36] and those where multiplication and reduction are interleaved [14,48]. The sequence of operations related to those implementations always contain the products $U[i] \cdot V[j]$ that were targeted in our attack. Hence, by applying the same approach as described in Section 4.3, it stays possible to distinguish the two following cases: "Case (1)" when the device processes two multiplications with independent operands and "Case (2)" when the device computes the multiplication of two related operands that jointly depend on a secret bit s. We recall hereafter some classical modular multiplication techniques.

Schoolbook Multiplication. This technique, also called *Long Integer Multiplication* (`LIM`), is a digit-by-digit multiplication algorithm where the products $U[i] \cdot V[j]$ are executed in the row order. An alternative approach has been introduced by Comba in [24]: it uses the same principle as the `LIM` but the products are taken in the column order.

Karatsuba-Ofman. This technique is very popular and is considered as one of the most efficient way to multiply two integers. If $t = 2n$, then U and V can be expressed as follows[10]:

$$U = U_H \cdot 2^{n\omega} + U_L \quad \text{and} \quad V = V_H \cdot 2^{n\omega} + V_L \ , \tag{12}$$

where U_H, V_H (resp. U_L, V_L) represent the n most significant ω-bit words of U and V (resp. the n least significant ω-bit words). The core idea of Karatsuba-Ofman multiplication is to process UV as follows:

$$U \cdot V = 2^{2n\omega}(U_H V_H) + 2^{n\omega}(U_H V_L + U_L V_H) + U_L V_L \ . \tag{13}$$

and

$$U_H V_L + U_L V_H = (U_H + V_L)(U_L + V_H) - U_H V_H - U_L V_L \ . \tag{14}$$

It may be checked that the processing of (13) and (14) may be done with 3 multiplications (instead of 4 with the `LIM`). By applying the idea recursively, the overall complexity is roughly reduced from t^2 to $t^{\log_2(3)}$. When such a multiplication algorithm is used, only the t final elementary multiplications $U[i]V[i]$, with $i \in [1, t]$ can be involved in a collision attack such as described in Section 4.3. This strongly decreases the efficiency of our attack.

Booth's Multiplication. The idea here is to rewrite the representation of the operands (for example by using a signed representation) in order to increase the number of zeroes in the latter. The advantage of this method is that it allows a faster multiplication. The multiplication is then performed as the `LIM`.

Montgomery's Multiplication. The principle of this method is to perform the modular multiplication using modular reductions easier to compute, by introducing an integer R, called the *radix*. R is defined such that $R = 2^{t\omega} > p$. Every element $x \in \mathbb{F}_p$ is then represented by $X = xR \bmod p$. This is called the

[10] If t is odd, it can be right-padded with a zero.

Montgomery representation of x. Assume two elements are given in their Montgomery representation U and V. To compute the Montgomery representation Z of their product, we first compute the standard multi-precision multiplication of U and V which is a number of size at most p^2. By applying Montgomery reduction to this result, we obtain Z. Thus, to multiply two elements in Montgomery representation, we only need to perform a single multi-precision multiplication followed by a Montgomery reduction. No division is needed.

In practice, this operation can be made more efficient by interleaving the multiplication and reduction steps. In our case, we will still be able to identify the elementary multiplications needed for the attack.

C Projective Coordinates

In Weierstrass Equation (3), points on elliptic curves are described in affine representation, namely using their (x, y)-coordinates. While it seems to be the simplest way to describe points over (E), addition and doubling formulas using affine coordinates require to compute the inverse of an element in $\mathbb{F}_p$, which is a very costly operation. This drawback led embedded systems developers to use other kinds of representations, such as for instance *the projective coordinates* that enable to perform point operations without requiring any field inversion. Moreover this type of representation avoids the need to resort to special treatment for the point at infinity. This is an advantage compared to the affine coordinates, since it prevents side-channel attacks that exploits the difference of representation between $\mathcal{O}$ and non-zero points.

To make it clear a point $P = (x, y)$ can be expressed in projective coordinates by a triplet $(X : Y : Z)$ such that $X = xZ$ and $Y = yZ$. Following this definition, point $(X : Y : Z)$ is the same as point $(\lambda X : \lambda Y : \lambda Z)$ for $\lambda \neq 0$.

Obviously other types of point representations share the same properties listed above with the projective coordinates. *Jacobian coordinates* or even the *Edwards'* ones are examples of such representations. They require a small number of elementary operations in order to add or double points on (E).