Side-Channel Analysis of Montgomery's Representation Randomization

Eliane Jaulmes(✉), Emmanuel Prouff, and Justine Wild

ANSSI, 51, Bd de la Tour-Maubourg, 75700 Paris 07 SP, France
{eliane.jaulmes,emmanuel.prouff,justine.wild}@ssi.gouv.fr

Abstract. Elliptic curve cryptography is today widely spread in embedded systems and the protection of their implementation against side-channel attacks has been largely investigated. At CHES 2012, a countermeasure has been proposed which adapts Montgomery's arithmetic to randomize the intermediate results during scalar point multiplications. The approach turned out to be a valuable alternative to the previous strategies based on hiding and/or masking techniques. It was argued to be specifically dedicated to hardware implementations and it aimed to defeat first-order side-channel attacks involving Pearson's correlation as distinguisher. In this paper however, we exhibit an important flaw in the countermeasure and we show, through various simulations, that it leads to efficient first-order correlation-based attacks.

1 Introduction

Elliptic Curves Cryptosystems (ECC) have been introduced by N. Koblitz [22] and V. Miller [29]. Their security relies on the hardness of the *discrete logarithm problem*. Elliptic curve based algorithms usually require keys far smaller than those involved in other public-key cryptosystems like RSA. This explains the current popularity of ECC and their involvement in a large variety of applications implemented over all kinds of devices: smart-cards, micro-controllers, and so on. Since such devices are widespread and in the hands of end-users, they are confronted to a wide range of threats. In particular, physical attacks need to be taken into account when assessing the overall security of the implementation. Thus, countermeasures are often conceived and implemented alongside the algorithms.

Physical attacks are traditionally divided into the two following families: *perturbation analysis* and *observation analysis*. The first one aims to modify the cryptosystem processing with any physical mean such as laser beams, clock jitter or voltage perturbation (*e.g.* fault injection attacks [12,13]). The attacker then learns information on the secret parameter by observing the response of the cryptosystem to this perturbation. Such attacks can be prevented by monitoring the device environment with captors and by verifying the result of the computation before output. The second family of attacks consists in measuring physical data during the algorithm execution and then in exploiting this information to recover the secret. Such leakage sources can be the power consumption or

A. Joux and A. Youssef (Eds.): SAC 2014, LNCS 8781, pp. 212–227, 2014.
DOI: 10.1007/978-3-319-13051-4_13

the electro-magnetic emanation. Among observation attacks, we can distinguish several categories. *Simple Power Analysis* [24] directly deduces the value of the secret from a single input processing (possibly averaged), while *Advanced Power Analysis* exploits observations for several algorithm inputs. The latter kind of attacks requires the choice of a *leakage model* to compare predictions based on key-hypotheses with the measured traces. The comparison is done with a statistical tool, also called *distinguisher*. For example, the well-known *Differential Power Analysis* (DPA) [25] uses the *difference of means* whereas the *Correlation Power Analysis* (CPA) [10] involves *Pearson's correlation coefficient*[1].

Countermeasures against observation analysis fall into two categories: *hiding* techniques aim at reducing the *Signal to Noise Ratio* (SNR) by increasing the noise or by equalizing the current in the circuit [35], while *masking* techniques consists in randomizing the sensitive computations. In practice, masking is always applied (possibly combined with hiding) since it provides strong security guaranty. Recently, Lee *et al.* [26] proposed a new efficient countermeasure to overcome first-order CPA[2] attacks. It assumes that the field operations are performed in the Montgomery domain [30] and consists in randomizing the Montgomery representation of the internal results (thus defining a so-called *Randomized Montgomery Domain*). This countermeasure has been considered as a very valuable alternative to the previous techniques because it avoids the need for scalar blinding (see *e.g.* [5]) and is much more efficient from a hardware implementation point of view.

Our work. In this paper, we show that the countermeasure proposed in [26] is flawed and can be efficiently broken by first-order CPA, even in presence of a large amount of noise in the measurements. After a presentation of the techniques proposed by Lee *et al.* in Sect. 2, the flaw is exhibited in Sect. 3.1. The attack is afterwards detailed in Sect. 3.2 and its efficiency is demonstrated in Sect. 4 thanks to simulations in various contexts. Finally, Sect. 5 analyses the experiments given in [26] and concludes with a short discussion about possible countermeasures.

2 On Randomized Implementations of Modular Operations

In this section, we recall some mathematical background on elliptic curve and the associated scalar multiplication. The reader may also refer to [3] for a more complete overview. Here, we will focus on the efficient implementation of the scalar multiplication in embedded devices. In particular we recall the use of the Montgomery domain for efficient modular operations [31].

[1] The need for a leakage model may be relaxed when using the so-called *collision attacks* [40] which look for colliding values during a computation. Such attacks compare only real traces to each other.

[2] A side channel attack is said to be *of first order* if it exploits the dependency between the mean of an instantaneous leakage and a function of the secret parameter. The original CPA attack in [10] is of first-order.

2.1 Background on Elliptic Curves and Montgomery Multiplication

Elliptic Curves. In this paper, we focus on *elliptic curves* E defined over a prime field $\mathbb{F}_p$ according to the following *short Weierstrass's Equation*:

$$E : y^2 = x^3 + ax + b, \tag{1}$$

where a and b are elements in $\mathbb{F}_p$ satisfying $4a^3 + 27b^2 \neq 0$. The set of rational points of E is denoted by $E(\mathbb{F}_p)$. It contains all the points whose coordinates $(x, y) \in \mathbb{F}_p^2$ satisfy (1). This set, augmented with a neutral element $\mathcal{O}$ called *point at infinity*, has an Abelian group structure for the following addition law: let $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ then the coordinates (x_3, y_3) of $P + Q$ satisfy;

$$x_3 = \lambda^2 - x_1 - x_2 \text{ and } y_3 = \lambda(x_1 - x_3) - y_1, \tag{2}$$

where λ equals $(y_2 - y_1)/(x_2 - x_1)$ if $P \neq Q$ and $(3x_1^2 + a)/(2y_1)$ otherwise. The *scalar multiplication* (`ECSM` for short) of a point $P \in E(\mathbb{F}_p)$ by a natural integer k is denoted by kP. It is the core operation of many cryptographic protocols such as `ECDSA` [22] and defining an efficient scalar multiplication arithmetic is hence a central issue (the interested reader is referred to [16] for a good overview). The point addition formula (2) is also central in elliptic curve implementations and several papers have been published on this subject [1,4,14,15,18,27,28,41]. They aim at proposing sequences of operations over $\mathbb{F}_p$ which are optimal according to some relations between the cost of a field inversion, the cost of an addition/subtraction and that of a multiplication. To defeat `SPA` attacks [5] at the arithmetic level, some works also try to propose sequences that stay unchanged whether (2) defines an addition ($P \neq Q$) or a doubling ($P = Q$). In this paper, we make no particular assumption on the type of `ECSM` algorithm nor on the sequence of operations which is used to process a point addition or a doubling. We indeed present a side-channel attack that does not exploit some particularity of the operations sequence or modus operandi but exploits information leakage during the manipulation of an intermediate result when the masking proposed in [26] is used. The latter type of result is likely to appear whatever the implementation choice.

Montgomery Domain. In order to efficiently perform the field operations involved in the point addition/doubling, developers often use the well-known Montgomery arithmetic [31]. This technique allows to replace the divisions occurring during the modular reduction by very efficient binary shifts. The only costly operation is the transformation to and from the Montgomery domain, but it is only done twice: at the beginning and at the end of the whole point operation (*e.g.* the scalar multiplication).

Let x represent an element in the prime field $\mathbb{F}_p \simeq \mathbb{Z}_p$ and let $R = 2^m$ be defined such that $2^m < p < 2^{m+1}$ (*i.e.* $m = \lfloor \log_2(p) \rfloor$). The value $\overline{x} = x \cdot R \bmod p$ is called the *Montgomery representation* of x and R is called the *Montgomery constant*. There is an isomorphism between $(\mathbb{Z}_p, +, \cdot)$ and the *Montgomery domain* $\mathrm{MD}(p) = (\{\overline{x}\}, +, \otimes)$, where $\otimes$ represents the *Montgomery multiplication* $\overline{x} \otimes \overline{y} = (\overline{x} \cdot \overline{y}) \cdot R^{-1} \bmod p$. The result of the multiplication corresponds to $\overline{xy}$, *i.e.* the Montgomery representation of $xy \bmod p$.

2.2 Randomized Montgomery Domain

Instead of using always $R = 2^m$, the authors of [26] propose to use a randomized Montgomery constant $r = 2^\lambda$ where λ is the Hamming Weight of a random m-bit value. The randomized Montgomery representation (RMR for short) of $x \in \mathbb{Z}_p$ is $x \cdot 2^\lambda \bmod p$. It is denoted $\widetilde{x}_\lambda$ or just $\widetilde{x}$ when there is no ambiguity on λ. The randomized Montgomery domain is denoted by $\mathtt{RMD}_\lambda(p)$. The multiplication in $\mathtt{RMD}_\lambda(p)$ works as follows: $\widetilde{x} \otimes_\lambda \widetilde{y} = (\widetilde{x} \cdot \widetilde{y})2^{-\lambda} \bmod p$. The result of the multiplication corresponds to $\widetilde{xy}$, *i.e.* the RMR of $xy \bmod p$. To ensure that the number of final subtractions stays upper-bounded by 1 (as in the classical Montgomery multiplication), the random power λ must be chosen in $[0..m]$.

In [26], the idea of RMR is applied to secure the implementation of an elliptic curve scalar multiplication against first-order side channel attack (*e.g.* CPA). The principle of these attacks is to observe the device behaviour during the processing of several scalar multiplications kP where the secret scalar $k \in \mathbb{N}$ stays unchanged and the public point P varies. Such attacks can be applied, for example, against some implementations of semi-static Diffie-Hellman key exchange, as found in the IEEE P1363 standard [17]. The main steps of the secure algorithm proposed in [26] are recalled hereafter, where the point P is assumed to belong to the set of rational points $E(\mathbb{F}_p)$ defined as in (1).

1. **Inputs:** a (public) point $P = (x_1, y_1) \in E(\mathbb{F}_p)$, a (secret) scalar $k \in \mathbb{N}$, a random number $\alpha \in [0..2^m - 1]$ with $m = \lfloor \log_2(p) \rfloor$.
2. **Conversion to RMD:** for $\lambda = \mathrm{HW}(\alpha)$, process
$$\widetilde{x_1} = x_1 \cdot 2^\lambda \bmod p$$
and
$$\widetilde{y_1} = y_1 \cdot 2^\lambda \bmod p.$$
Also convert the curve parameters (a, b) into $\mathtt{RMD}_\lambda(p)$. The point with coordinates $(\widetilde{x_1}, \widetilde{y_1})$ is denoted by $\widetilde{P}$.
3. **Elliptic Curve Scalar Multiplication (ECSM):** in $\mathtt{RMD}_\lambda(p)$, process
$$\widetilde{Q} = (\widetilde{x_2}, \widetilde{y_2}) = k\widetilde{P}. \tag{3}$$
4. **Conversion to Integer Domain:** process
$$x_2 = \widetilde{x_2} \otimes_\lambda 1 = \widetilde{x_2} \cdot 2^{-\lambda} \bmod p$$
and
$$y_2 = \widetilde{y_2} \otimes_\lambda 1 = \widetilde{y_2} \cdot 2^{-\lambda} \bmod p$$
5. **Output:** $Q = (x_2, y_2) = kP$

The Elliptic Curve Scalar Multiplication (ECSM) may be done with any algorithm (*e.g.* [31]). The authors of [26] do not recommend any particular one, even if the resistance tests reported in their paper are applied against an ECSM based on Montgomery Ladder (see Algorithm 5 in Appendix A). Similarly, our attack

described in the next section does not exploit any particular feature of the ECSM and thus applies independently from the choice of algorithm. To allow for comparisons with [26] we however chose to target an ECSM based on Montgomery Ladder in our attack simulations.

3 Our Attack

3.1 Core Idea

The goal of our attack is to recover the bits of k one after another during the processing of ECSM in the randomized Montgomery domain (*i.e.* the processing of (3)). In this section, we detail how the second MSB (respectively the LSB) of k is recovered[3]. Once this bit is obtained, the attack can be applied similarly on the other bits from left to right (respectively right to left).

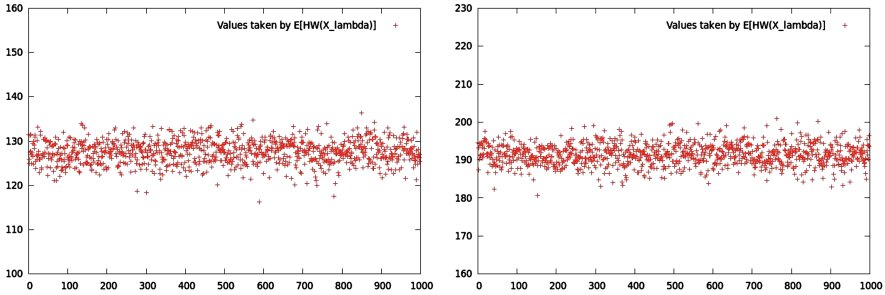
We denote by $R(\widetilde{u}, \widetilde{v})$ the value of an intermediate point during the processing of ECSM in the randomized Montgomery domain. We assume that the coordinates of the point R depend on a small sub part of k (*e.g.* a bit). This part is denoted by s. It can for instance correspond to the result of the second point operation in ECSM with Montgomery Ladder (see (8)) or to the result of the third point operation in ECSM with double-and-add always method (see (6)–(7)). Our side channel attack targets the manipulation of the first coordinate $\widetilde{u}$ of this point by the device: by assumption, it satisfies $\widetilde{u} = f(\widetilde{x}, \widetilde{y}, s)$ where we recall that $(\widetilde{x}, \widetilde{y})$ denotes the RMR coordinates of the input point P and where f is a known function that depends on the curve parameters and the algorithm used to process ECSM (see Appendix A for examples). To simplify the presentation, we assume that s is reduced to a single bit of k (the MSB or the LSB) but our attack straightforwardly applies to higher values (the only restriction is that the upper bound must be small enough to allow for an exhaustive test of all the elements). Our attack is based on the following statement which essentially means that a RMR representation leaks information on the un-blinded coordinate and that this information can be exploited by a *first-order* side-channel attack:

Statement: the function $f : x \mapsto \mathbb{E}\big[\mathrm{HW}(\widetilde{X}_\lambda) \mid X = x\big]$ is not constant.

Remark 1. Taking into account the specificities of the randomized Montgomery representation recalled in Sect. 2.2, the mean is computed over the random variable λ defined such that $\lambda = \mathrm{HW}(\alpha)$ with α having a uniform distribution on $[0..2^m - 1]$.

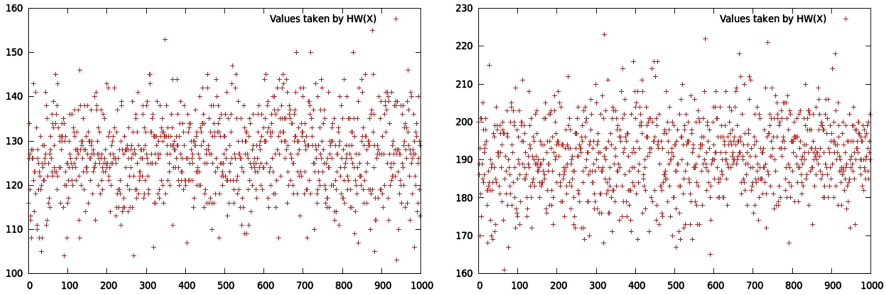
To argue on the statement above, we evaluated f on 1000 different values x for $m = 256$ and $m = 384$ (these parameters are recommended for instance by the American National Security Agency when ECSM is used to process an ECDSA). The results are plotted in Fig. 1(a) and (b). For comparison, we also

[3] To easy the explanation of the attack against left-to-right implementations of ECSM, we make the classical assumption that the MSB of k equals 1.



(a) Curves of size 256 bits (b) Curves of size 384 bits

Fig. 1. Values of $f : x \mapsto \mathbb{E}\big[\mathrm{HW}(\widetilde{X}) \mid X = x\big]$ for 1000 random values x

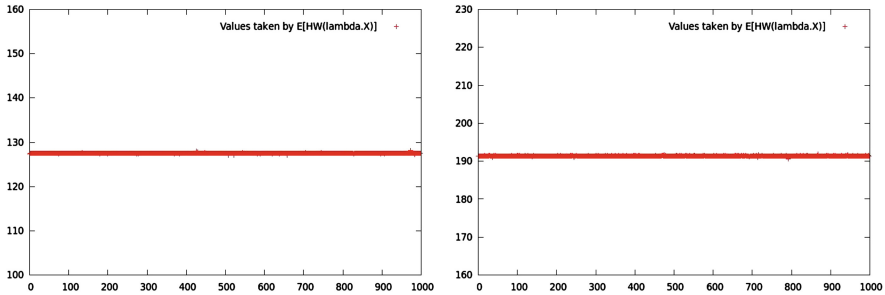


(a) Curves of size 256 bits (b) Curves of size 384 bits

Fig. 2. Values of $f : x \mapsto \mathrm{HW}(x)$ for 1000 random values x

plotted in Fig. 2(a) and (b) the value $\mathrm{HW}(x)$ for 1000 values x in $[0..p)$, which corresponds to the case of non blinded values. Finally, in Fig. 3(a) and (b) we plotted $\mathbb{E}\big[\mathrm{HW}(\lambda \cdot X \bmod p) \mid X = x\big]$ where λ is a random uniformly distributed 16-bit value. The latter corresponds to the case where `ECSM` is protected by blinding the projective coordinates of the point P.

As expected, we can see in Fig. 2(a) and (b) that the Hamming weight of x varies with x when no blinding is involved; this implies that a first-order `CPA` is possible if x is sensitive. On the contrary, Fig. 3(a) and (b) show that coordinate blinding cancels any leakage on x since the average Hamming weight of the blinded coordinate is almost constant (even in our case where we limited λ to 16-bits words which can be considered as too limited in practice). Between the two previous extreme cases, Fig. 1(a) and (b) show that $f(x)$ varies with x (not with a high variance as in Fig. 2(a) and (b) but visibly much more than in Fig. 3(a) and (b)): this implies that the countermeasure in [26] can be attacked with a first-order `CPA` involving f to process the predictions on $\widetilde{x}$. In the following section, we detail the latter attack.



(a) Curves of size 256 bits (b) Curves of size 384 bits

Fig. 3. Values of $f : x \mapsto \mathbb{E}\big[\mathrm{HW}(\lambda \cdot X \bmod p) \mid X = x\big]$ for 1000 random values x

3.2 Attack Description

As explained in the previous section, our attack recovers the secret piece by piece. Here, we detail how it allows for the recovery of the second `MSB` or the `LSB` s of the secret scalar k (the principle can then be repeated to recover the other bits one after another). The intermediate result/point $R(\widetilde{u}, \widetilde{v})$ exploited by the attack differs with the algorithm used to process the scalar multiplication. We give some examples below but our first-order side-channel attack also applies in other contexts (*e.g.* against atomic implementations or implementations applying the *window* principle [21]) as long as the point is blinded with the `RMR` representation proposed in [26] and the scalar is not blinded.

Several scalar multiplication algorithms are recalled in Appendix A. We recall that, for left-to-right versions, we make the (classical) assumption that the `MSB` equals 1. Our attack targets the manipulation of the first coordinate $\widetilde{u}$ of the intermediate result/point $R(\widetilde{u}, \widetilde{v})$ corresponding to:

– [**left-to-right double-and-add** `ECSM`] the second point operation

$$R = 3P + (1 - s)P. \tag{4}$$

– [**right-to-left double-and-add** `ECSM`] the first point operation

$$R = P + (1 - s)P. \tag{5}$$

– [**left-to-right double-and-add-always** `ECSM`] the third point operation

$$R = 2(P + sP). \tag{6}$$

– [**right-to-left double-and-add-always** `ECSM`] the third point operation

$$R = 2P + sP. \tag{7}$$

– [**Montgomery Ladder** `ECSM`] the second point operation

$$R = 2(P + sP). \tag{8}$$

– [**Joye** ECSM] the second point operation

$$R = P + (1 - s)P. \tag{9}$$

Following the classical outlines of a first-order side-channel attack, our attack starts by the observation of the device behaviour for several executions of the algorithm ECSM parametrized by different public inputs P but a same secret scalar k. Note that only the part of the observation corresponding to the manipulation of the coordinate $\widetilde{u}$ of R is used in the attack described hereafter. Since the i^{th} observation (*e.g.* the power consumption or the electromagnetic emanation) is algorithmically related to the i^{th} input point P_i and the secret bit s, it is denoted by $\mathcal{L}(s, P_i)$ in the following. At the end of this measurement step of the attack, the adversary is assumed to be provided with a sample of pairs $(P_i, \mathcal{L}(s, P_i))_i$. The size of this sample is denoted by N.

To underline the functional dependency between $\widetilde{u}$ and (s, P_i), we use the notation $\widetilde{u}(s, P_i)$ in the following. For testing an hypothesis $\hat{s}$ on s, the attack continues with the computation of the values $u(\hat{s}, P_i)$ for $i \in [1..N]$. These values correspond to the *unmasked* version of the u-coordinates under the hypothesis $\hat{s} = s$. Then applying the strategy already used in [6,36] and argued in [38], we choose a *device model* m (usually the Hamming weight) and we compute the sample of predictions $(h_i)_{i \leqslant N}$ defined such that:

$$h_i(\hat{s}) = \mathbb{E}_\lambda\big[\mathsf{m}(u(\hat{s}, P_i) \cdot 2^\lambda \bmod p)\big] = \sum_{i=1}^{m} \mathsf{m}(u(\hat{s}, P_i) \cdot 2^i \bmod p) \times \mathrm{p}[\lambda = i].$$

By construction λ is defined as the Hamming weight of an m-bit random element. Its distribution is therefore binomial with parameters m and $1/2$ and the equation above can be developed as follows:

$$h_i(\hat{s}) = \frac{1}{2^m} \sum_{i=1}^{m} \mathsf{m}(u(\hat{s}, P_i) \cdot 2^i \bmod p) \times \binom{m}{i}. \tag{10}$$

Eventually, the absolute value of the correlation $\rho_{\hat{s}}$ between the samples $(h_i(\hat{s}))_i$ and $(\mathcal{L}(s, P_i))_i$ is computed for $\hat{s} \in \{0, 1\}$ and the attack returns the hypothesis with the greatest correlation.

4 Simulations

Setting. In this section, we assume that the ECSM is implemented on a 32-bit architecture and according to the Montgomery ladder (Algorithm 5 in Appendix A). In such environment which corresponds to a classical context, data will only be manipulated through 32-bit registers. Without loss of generality, we hence assume that the leakage exploited in our attacks is the Hamming weight of the 32 least significant bits of $\widetilde{u}$ (namely $\mathrm{HW}(\widetilde{u} \bmod 2^{32})$) instead of the Hamming weight of the whole 256 or 384 bit value[4]. The described attack aims at

[4] The attack applies similarly whether the attacker chose any 32-bit part of the targeted value.

recovering the most significant bit s of the secret k and, according to (8), the observations are hence assumed to be related to the manipulation of the coordinate $\widetilde{u}$ of the point $R(\widetilde{u}, \widetilde{v}) = 2(P + sP)$. The latter observations are simulated in the classical Hamming weight model with Gaussian Noise. Namely, the leakage observation $\mathcal{L}(s, P_i)$ corresponding to the i^{th} scalar multiplication is simulated such that:

$$\mathcal{L}(s, P_i) = \mathrm{HW}(\widetilde{u} \bmod 2^{32}) + \mathcal{N}, \tag{11}$$

where $\widetilde{u} = u \cdot 2^{\lambda_i} \bmod p$ is the first coordinate of $2(P_i + sP_i)$ represented in the **randomized** Montgomery domain $\mathtt{RMD}_{\lambda_i}(p)$ (with λ_i generated at random in its definition set) and where $\mathcal{N}$ is a Gaussian random variable with mean 0 and standard deviation σ.

Since the leakage is assumed to satisfy (11), the model function m in Eq. (10) has been simply chosen to be the Hamming weight of the 32 least significant bits of the input. The predictions $h_i(\hat{s})$ associated to the binary hypothesis $\hat{s}$ on s hence satisfy:

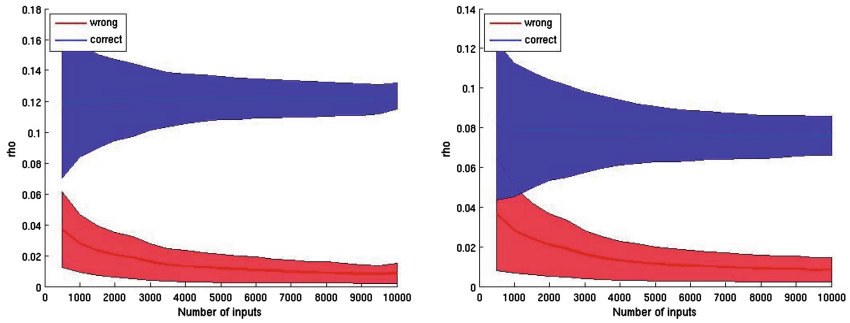
$$h_i(\hat{s}) = \frac{1}{2^m} \sum_{i=1}^{m} \mathrm{HW}((u \cdot 2^i \bmod p) \bmod 2^{32}) \times \binom{m}{i}, \tag{12}$$

where u is the first coordinate of $2(P_i + \hat{s}P_i)$ in the standard integer domain and where m equals 256 or 384.

To sum-up, the simulations reported in this section are split in two parts. The first part aims to show that the `CPA` (*i.e.* the correlation coefficient) succeeds in distinguishing the correct hypothesis from the wrong one. The second part serves to estimate the success rate in recovering the first secret bit in a Montgomery ladder implementation of `ECSM`. The latter success rate is estimated for different noise levels σ and different number of observations.

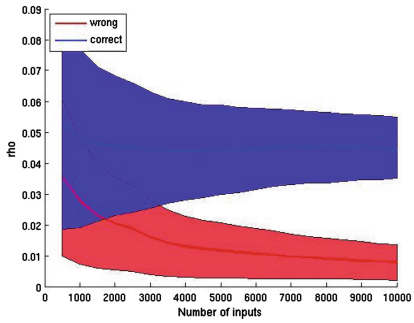
Pearson Correlation Coefficient. First, we are comparing the correlation values for the correct and the wrong predictions. Following the attack description in Sect. 3.2, we compute the set of hypotheses $(h_i(\hat{s} = 1))_{i \leqslant N}$ according to (12), and the two sets of leakages $(\mathcal{L}(1, P_i))_{i \leqslant N}$ and $(\mathcal{L}(0, P_i))_{i \leqslant N}$, according to (11), where we recall that N represents the total number of randomly chosen points P_i. The first set corresponds to leakages that match the chosen hypotheses and the second one to leakages that do not match the hypotheses. These experiments have been repeated for different values of $N \in [500..10000]$ and different noise standard deviation $\sigma \in [0..30]$. Then, we compute the two correlation coefficients $\rho_{\text{correct}} = \rho((h_i(\hat{s} = 1))_i, (\mathcal{L}(1, P_i))_i)$ and $\rho_{\text{wrong}} = \rho((h_i(\hat{s} = 1))_i, (\mathcal{L}(0, P_i))_i)$. The obtained correlation values are then averaged over the execution of 100 such attacks and the standard deviation of the correlation values is also computed to measure how much the mean is informative.

Results of this first set of experiments are presented in Fig. 4. It represents the average and the standard deviation cone for the correlation coefficients obtained with the correct and wrong predictions. It may be seen that the correlation

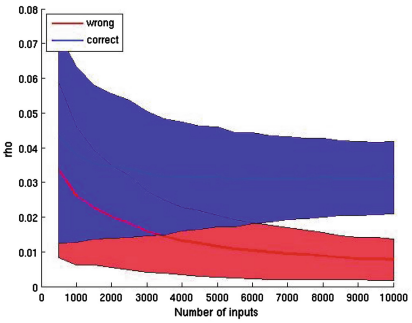


(a) Noise = 0

(b) Noise = 10



(c) Noise = 20



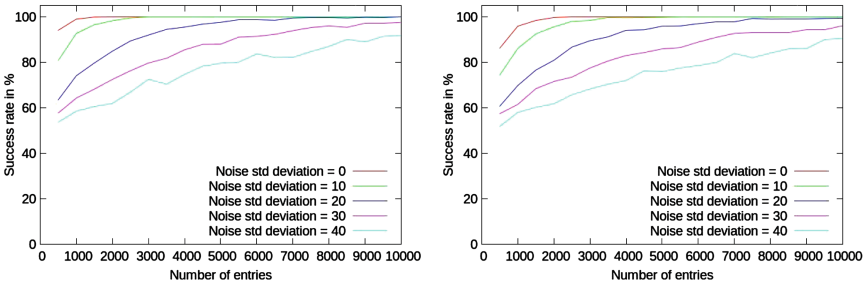
(d) Noise = 30

Fig. 4. Correlation for 256-bit curve attacking with known messages

coefficient allows us to distinguish a correct key bit hypothesis from a wrong one with high probability as long as the number of observations satisfies:

$$\begin{cases} N \geq 500 & \text{if } \sigma = 0 \\ N \geq 1500 & \text{if } \sigma = 10 \\ N \geq 3000 & \text{if } \sigma = 20 \\ N \geq 6000 & \text{if } \sigma = 30 \end{cases}.$$

Success Rate. We then proceed to test the attack efficiency. For such a purpose, we randomly choose the secret s we are trying to recover. We then simulate the leakage observations according to (11) for N pairs (P_i, λ_i) of values generated at random in their respective definition set. This step provides us with a set $(\mathcal{L}(s, P_i))_{i \leqslant N}$ playing the role of the registered traces in a real attack scenario. We then compute the two sets of predictions $(h_i(\hat{s} = 0))_{i \leqslant N}$ and $(h_i(\hat{s} = 1))_{i \leqslant N}$, and we process the absolute value of the two corresponding correlation coefficients $\rho_0 = \rho((\mathcal{L}(s, P_i)_i, (h_i(\hat{s} = 0))_i)$ and $\rho_1 = \rho((\mathcal{L}(s, P_i)_i, (h_i(\hat{s} = 1))_i)$. The prediction $\hat{s}$ such that $|\rho_{\hat{s}}|$ has the highest value is set as the most likely one. If $\hat{s} = s$ the attack succeeds, else it fails. In our experiments we repeated each attack 1000 times with different random values to build a success rate.



(a) Curves of size 256 bits (b) Curves of size 384 bits

Fig. 5. Success rate for several noise values

Figure 5 represents the success rate in percentage when trying to guess the bit s. In abscissa is the number of different inputs used for the computation of the correlation coefficient. Several noise standard deviation values have been used. It may be checked that even for $\sigma = 40$ (*i.e.* $\texttt{SNR} = 0,005$) the attack succeeds with probability greater than 80 % if the number N of observations used by the attacker is greater than 5500 for curves of size 256 bits, respectively greater than 7000 for 384-bit curves.

5 Analysis and Conclusion

In this paper, we have argued that the countermeasure proposed in [26] is flawed and does not defeat first-order CPA. Actually, through attack simulations conducted under reasonable and classical assumptions on the execution environment we have shown that a CPA is likely to be very efficient even when the noise is huge. The problem identified in the countermeasure under study is that the distribution of the masking values λ is binomial (and not uniform). This choice, which has been done for efficiency reason, is in fact dramatic from a security point of view. It must be mentioned that the authors of [26] tested a schoolbook CPA against their implementation and used the failure of this attack to argue on the resistance of their countermeasure. As we have shown here, the failure of the CPA performed in [26] is not a consequence of the countermeasure quality but of a wrong attack parametrization. By adapting to a first-order context the argumentation given in [38] for second-order attacks, we were indeed able to use a much better parametrization for the CPA. Our work demonstrates once again that countermeasures must not be only validated by performing some *ad-hoc* attacks (even classical) but must be formally analysed, for instance by following the approaches proposed in [7,9,37].

Concerning the proposed countermeasure, a possible patching could consist in generating λ uniformly in $[0..m]$ but a careful security analysis is needed to assess on the pertinence of this patch. The proposal of Dupaquis and Venelli in [8] may offer an interesting alternative in the case where efficiency is required. Otherwise, well-studied approaches such as classical exponent and message blinding seem to offer better security garanties.

A Examples of Algorithms for Elliptic Curve Scalar Multiplication

We recall hereafter two basic algorithms (see Algorithms 1 and 2) to calculate the scalar multiplication either from left-to-right or from right-to-left[5].

Algorithm 1. Left-to-Right Binary ECSM

Input : a point P on E and a secret scalar $k = (1, k_{n-2}, \cdots, k_0)_2$
Output: the point $Q = kP$

$R_0 \leftarrow P$
$R_1 \leftarrow P$
for $i = n-2$ **to** 0 **do**
  $R_0 \leftarrow 2R_0$
  if $k_i = 1$ **then**
    $R_0 \leftarrow R_0 + R_1$
return R_0

Algorithm 2. Right-to-Left Binary ECSM

Input : a point P on $E(\mathbb{F}_p)$ and a secret scalar $k = (k_{n-1}, \cdots, k_0)_2$
Output: the point $Q = kP$

$R_0 \leftarrow \mathcal{O}$
$R_1 \leftarrow P$
for $i = 0$ **to** $n-1$ **do**
  if $k_i = 1$ **then**
    $R_0 \leftarrow R_0 + R_1$
  $R_1 \leftarrow 2R_1$
return R_1

The previous algorithms are simple and relatively efficient (n additions and $n/2$ doublings in average). However, they are not regular and can hence induce an information leakage exploitable by SPA. As for instance explained in [39], regularity can be achieved by using unified formulae for point addition and point doubling [11] or by the mean of side-channel *atomicity* whose principle is to build point addition and point doubling algorithms from the same atomic pattern of field operations [2]. Another possibility is to render the scalar multiplication algorithm itself regular, independently of the field operation flows in each point operation. Namely, one designs a scalar multiplication with a constant flow of point operations. This approach was first followed by Coron in [5] who proposed to perform a dummy addition in the binary algorithm loop whenever the scalar

[5] To simplify the attacks description, and because it has no impact on their feasibility, it is assumed for the left-to-right versions that the most significant bit of k always equals 1, *i.e.* that the bit-length is exactly n.

bit equals 0 or not. The obtained *double-and-add-always* algorithm (see Algorithms 3 and 4) performs a point doubling and a point addition at every loop iteration and the scalar bits are no more distinguishable by SPA.

Algorithm 3. Left-to-Right double-and-add always ECSM [5]

Input : a point P on $E(\mathbb{F}_p)$ and a secret scalar $k = (1, k_{n-2}, \cdots, k_0)_2$
Output: the point $Q = kP$

$R_0 \leftarrow P$
for $i = n - 2$ **to** 0 **do**
  $R_0 \leftarrow 2R_0$
  $R_1 \leftarrow R_0 + P$
  $R_0 \leftarrow R_{k_i}$
return R_0

Algorithm 4. Right-to-Left double-and-add always ECSM [19]

Input : a point P on $E(\mathbb{F}_p)$ and a secret scalar $k = (k_{n-1}, \cdots, k_0)_2$
Output: the point $Q = kP$

$R_0 \leftarrow P$
$R_1 \leftarrow \mathcal{O}$
for $i = 0$ **to** $n - 1$ **do**
  $R_2 \leftarrow R_0 + R_1$
  $R_0 \leftarrow 2R_0$
  $R_1 \leftarrow R_{1+k_i}$
return R_1

Other regular binary algorithms exist such as the *Montgomery ladder* [32] (see Algorithm 5) and the double-and-add algorithm proposed by Joye in [20] (see Algorithm 6). These algorithms which are recalled hereafter not only counteract SPA but also some fault attacks (such that the *safe-error* ones).

Algorithm 5. Montgomery Ladder ECSM [32]

Input : a point P on $E(\mathbb{F}_p)$ and a secret scalar $k = (1, k_{n-2}, \cdots, k_0)_2$
Output: the point $Q = kP$

$R_0 \leftarrow P$
$R_1 \leftarrow 2P$
for $i = n - 2$ **to** 0 **do**
  $b \leftarrow k_i$
  $R_{1-b} \leftarrow R_0 + R_1$
  $R_b \leftarrow 2R_b$
return R_0

Algorithm 6. Joye double-and-add always ECSM [20]

Input : a point P on $E(\mathbb{F}_p)$ and a secret scalar $k = (k_{n-1}, \cdots, k_0)_2$
Output: the point $Q = kP$

```
R_0 ← O
R_1 ← P
for i = 0 to n − 1 do
    b ← k_i
    R_{1−b} ← 2R_{1−b} + R_b
return R_0
```

References

1. Baldwin, B., Goundar, R.R., Hamilton, M., Marnane, W.P.: Co-z ecc scalar multiplications for hardware, software and hardware-software co-design on embedded systems. J. Crypt. Eng. **2**(4), 221–240 (2012)
2. Chevallier-Mames, B., Ciet, M., Joye, M.: Low-cost solutions for preventing simple side-channel analysis: side-channel atomicity. IEEE Trans. Comput. **53**(6), 760–768 (2004)
3. Cohen, H., Frey, G. (eds.): Handbook of Elliptic and Hyperelliptic Curve Cryptography. CRC Press, Boca Raton (2005)
4. Cohen, H., Miyaji, A., Ono, T.: Efficient elliptic curve exponentiation using mixed coordinates. In: Ohta, K., Pei, D. (eds.) ASIACRYPT 1998. LNCS, vol. 1514, pp. 51–65. Springer, Heidelberg (1998)
5. Coron, J.-S.: Resistance against differential power analysis for elliptic curve cryptosystems. In: Koç, Ç.K., Paar, C. [23], pp. 292–302
6. Coron, J.-S., Prouff, E., Roche, T.: On the use of shamir's secret sharing against side-channel analysis. In: Mangard, S. (ed.) CARDIS 2012. LNCS, vol. 7771, pp. 77–90. Springer, Heidelberg (2013)
7. Duc, A., Dziembowski, S., Faust, S.: Unifying leakage models: from probing attacks to noisy leakage. In: Nguyen, P.Q., Oswald, E. [34], pp. 423–440
8. Dupaquis, V., Venelli, A.: Redundant modular reduction algorithms. In: Prouff, E. (ed.) CARDIS 2011. LNCS, vol. 7079, pp. 102–114. Springer, Heidelberg (2011)
9. Durvaux, F., Standaert, F.-X., Veyrat-Charvillon, N.: How to certify the leakage of a chip? In: Nguyen, P.Q., Oswald, E. [34], pp. 459–476
10. Brier, E., Clavier, C., Olivier, F.: Correlation power analysis with a leakage model. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 16–29. Springer, Heidelberg (2004)
11. Brier, É., Joye, M.: Weierstraß elliptic curves and side-channel attacks. In: Naccache, D., Paillier, P. [33], pp. 335–345
12. Giraud, C., Thiebeauld, H.: A survey on fault attacks. In: Quisquater, J.-J., Paradinas, P., Deswarte, Y., El Kalam, A.A. (eds.) CARDIS 2004. LNCS, vol. 153, pp. 159–176. Springer, Heidelberg (2004)
13. Giraud, C., Thiebeauld, H.: Basics of fault attacks. In: Breveglieri, L., Koren, I. (eds.) Workshop on Fault Diagnosis and Tolerance in Cryptography - FDTC'04, pp. 343–347. IEEE Computer Society (2004)
14. Goundar, R.R., Joye, M., Miyaji, A.: Co-Z addition formulæ and binary ladders on elliptic curves. In: Mangard, S., Standaert, F.-X. (eds.) CHES 2010. LNCS, vol. 6225, pp. 65–79. Springer, Heidelberg (2010)

15. Goundar, R.R., Joye, M., Miyaji, A., Rivain, M., Venelli, A.: Scalar multiplication on weierstraß elliptic curves from co-z arithmetic. J. Crypt. Eng. **1**(2), 161–176 (2011)
16. Hankerson, D., Menezes, A.J., Vanstone, S.: Guide to Elliptic Curve Cryptography. Springer Professional Computing Series. Springer, New York (2003)
17. IEEE Std 1363–2000. IEEE Standard Specifications for Public Key Cryptography. IEEE Computer Society, January 2000
18. Itoh, K., Takenaka, M., Torii, N., Temma, S., Kurihara, Y.: Fast implementation of public-key cryptography ona DSP TMS320C6201. In: Koç, Ç.K., Paar, C. [23], pp. 61–72
19. Izu, T., Takagi, T.: A fast parallel elliptic curve multiplication resistant against side channel attacks. In: Naccache, D., Paillier, P. [33], pp. 280–296
20. Joye, M.: Highly regular right-to-left algorithms for scalar multiplication. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 135–147. Springer, Heidelberg (2007)
21. Knuth, D.E.: The Art of Computer Programming, vol. 2, 3rd edn. Addison Wesley, Reading (1988)
22. Koblitz, N.: Elliptic curve cryptosystems. Math. Comput. **48**(177), 203–209 (1987)
23. Koç, Ç.K., Paar, C. (eds.): CHES 1999. LNCS, vol. 1717. Springer, Heidelberg (1999)
24. Kocher, P.C.: Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 104–113. Springer, Heidelberg (1996)
25. Kocher, P.C., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999)
26. Lee, J.-W., Chung, S.-C., Chang, H.-C., Lee, C.-Y.: An efficient countermeasure against correlation power-analysis attacks with randomized montgomery operations for DF-ECC processor. In: Prouff, E., Schaumont, P. (eds.) CHES 2012. LNCS, vol. 7428, pp. 548–564. Springer, Heidelberg (2012)
27. Longa, P., Miri, A.: New composite operations and precomputation scheme for elliptic curve cryptosystems over prime fields. In: Cramer, R. (ed.) PKC 2008. LNCS, vol. 4939, pp. 229–247. Springer, Heidelberg (2008)
28. Meloni, N.: New point addition formulae for ECC applications. In: Carlet, C., Sunar, B. (eds.) WAIFI 2007. LNCS, vol. 4547, pp. 189–201. Springer, Heidelberg (2007)
29. Miller, V.S.: Use of elliptic curves in cryptography. In: Williams, H.C. (ed.) CRYPTO 1985. LNCS, vol. 218, pp. 417–426. Springer, Heidelberg (1986)
30. Montgomery, P.L.: Evaluating recurrences of form $X_{m+n} = f(X_m, X_n, X_{m-n})$ via Lucas chains 1983, Revised (1992). ftp.cwi.nl:/pub/pmontgom/Lucas.ps.gz
31. Montgomery, P.L.: Modular multiplication without trial division. Math. Comput. **44**(170), 519–521 (1985)
32. Montgomery, P.L.: Speeding the pollard and elliptic curve methods of factorization. Math. Comput. **48**, 243–264 (1987)
33. Naccache, D., Paillier, P. (eds.): PKC 2002. LNCS, vol. 2274. Springer, Heidelberg (2002)
34. Nguyen, P.Q., Oswald, E. (eds.): EUROCRYPT 2014. LNCS, vol. 8441. Springer, Heidelberg (2014)
35. Popp, T., Mangard, S.: Masked dual-rail pre-charge logic: DPA-resistance without routing constraints. In: Rao, J.R., Sunar, B. (eds.) CHES 2005. LNCS, vol. 3659, pp. 172–186. Springer, Heidelberg (2005)

36. Prouff, E., McEvoy, R.: First-order side-channel attacks on the permutation tables countermeasure. In: Clavier, C., Gaj, K. (eds.) CHES 2009. LNCS, vol. 5747, pp. 81–96. Springer, Heidelberg (2009)
37. Prouff, E., Rivain, M.: Masking against side-channel attacks: a formal security proof. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 142–159. Springer, Heidelberg (2013)
38. Prouff, E., Rivain, M., Bévan, R.: Statistical analysis of second order differential power analysis. IEEE Trans. Comput. **58**(6), 799–811 (2009)
39. Rivain, M.: Fast and regular algorithms for scalar multiplication over elliptic curves. IACR Cryptology ePrint Arch. **2011**, 338 (2011)
40. Schramm, K., Wollinger, T., Paar, C.: A new class of collision attacks and its application to DES. In: Johansson, T. (ed.) FSE 2003. LNCS, vol. 2887, pp. 206–222. Springer, Heidelberg (2003)
41. Venelli, A., Dassance, F.: Faster side-channel resistant elliptic curve scalar multiplication. In: Kohel, D., Rolland, R. (eds.) Arithmetic, Geometry, Cryptography and Coding Theory 2009, Contemporary Mathematics, vol. 521, pp. 29–40. American Mathematical Society (2010)