Side-Channel Analysis on Blinded Regular Scalar Multiplications

Benoit Feix[1] and Mylène Roussellet[2⋆] and Alexandre Venelli[3⋆]

[1] UL Security Transactions, UK Security Lab
benoit.feix@ul.com
[2] Gemalto, La Ciotat, France
mylene.roussellet@gemalto.com
[3] Thalès Communications & Security, Toulouse, France
alexandre.venelli@thalesgroup.com

Abstract. We present a new side-channel attack path threatening state-of-the-art protected implementations of elliptic curves embedded scalar multiplications. Regular algorithms such as the double-and-add-always and the Montgomery ladder are commonly used to protect the scalar multiplication from simple side-channel analysis. Combining such algorithms with scalar and/or point blinding countermeasures lead to scalar multiplications protected from all known attacks. Scalar randomization, which consists in adding a random multiple of the group order to the scalar value, is a popular countermeasure due to its efficiency. Amongst the several curves defined for usage in elliptic curves products, the most used are those standardized by the NIST. As observed in several previous publications, the modulus, hence the orders, of these curves are sparse, primarily for efficiency reasons. In this paper, we take advantage of this specificity to present new attack paths which combine vertical and horizontal side-channel attacks to recover the entire secret scalar in state-of-the-art protected elliptic curve implementations.

Keywords: Elliptic curves, Scalar multiplication, Side-channel analysis, Correlation analysis

1 Introduction

Elliptic Curve Cryptography (ECC) has become a very promising branch of cryptology. Since its introduction by Miller [25] and Koblitz [22] numerous studies have offered a rich variety of implementation methods to perform efficient and tamper resistant scalar multiplication algorithms in embedded products. Many standardized protocols like the *Elliptic Curve Digital Signature Algorithm* (ECDSA) [29] or the *Elliptic Curve Diffie-Hellman* (ECDH) are more and more used in payment and identity products. They have the strong advantage today to require significantly smaller parameters and key sizes than the well-known

⋆ This work was carried out when the author was with INSIDE Secure.

RSA [30] and Diffie-Hellman [15] cryptosystems. Most industrial ECC applications use elliptic curves defined in international standards [29, 32, 5]. These curves were generated with efficiency and security advantages for different classical security levels.

Besides these efficiency requirements in embedded environment, developers must also prevent their products from physical attacks. These techniques are split in two categories namely the *Side-Channel Analysis* (SCA) and the *Fault Analysis* (FA). In this paper, we use the full spectrum of *Side-Channel Analysis* namely classical *Vertical Correlation attacks* [7], *Horizontal Correlation attacks* [12], *Vertical Collision-Correlation* [38, 27] and *Horizontal Collision-Correlation* [13, 1].

A recent paper at Indocrypt 2013 from Bauer *et al.* [2] presented a new side-channel attack, combining vertical and horizontal techniques, on a standard RSA blinded exponentiation when the public exponent value is 3. Based on the same observation, we design new side-channel attack paths on regular scalar multiplication algorithms with blinded scalar implementations for most standardized curves. We present vertical and horizontal attacks with known and unknown input point values that successfully recover the whole secret scalar.

Our proposed attack strategy. Our attack paths consist of three steps. First, the attacker uses the fact that the scalar blinding does not mask a large part of the secret. This side-channel vulnerability can be exploited vertically, *i.e.* using several execution traces. The attacker will recover the middle part of the secret. In a second step, he needs to recover the random value used for each scalar blinding. This part is performed horizontally, *i.e.* each random will be recovered using only one trace. The already recovered part of the secret in the first step can provide more side-channel information to exploit for the attacker. This step allows to recover the most significant part of the scalar. Finally, the third step consists in retrieving the least significant part of the scalar. Using the already recovered random values of each traces and the middle part of the secret, the attacker can perform a vertical attack.

Roadmap. The paper is organized as follows. Section 2 reminds basics on elliptic curve cryptography and embedded scalar multiplication. We also detail the classical side-channel countermeasures and explain the side-channel attack knowledge necessary for a good understanding of the rest of the paper. In Section 3, we describe our first attack that defeats a regular implementation when the secret scalar is blinded but not the input point. Section 4 extends our attack techniques to the unknown (or randomized) input point case. To illustrate our attacks efficiency, we present experimental results on simulated side-channel traces in Section 5. Discussion on countermeasures is done in Section 6. We finally conclude our paper in Section 7.

2 Preliminaries

2.1 Background on Elliptic Curves

Let $\mathbb{F}_p$ be a finite field of characteristic $\neq 2, 3$. Consider an elliptic curve E over $\mathbb{F}_p$ given by the short Weierstraß equation $y^2 = x^3 + ax + b$, where $a, b \in \mathbb{F}_p$ and with discriminant $\Delta = -16(4a^3 + 27b^2) \neq 0$. The set of points on an elliptic curve form a group under the chord-and-tangent law. The neutral element is the point at infinity $\boldsymbol{O}$. Let $\boldsymbol{P} = (x_1, y_1)$ and $\boldsymbol{Q} = (x_2, y_2)$ be two affine points on $E(\mathbb{F}_p)$, their sum $\boldsymbol{R} = \boldsymbol{P} + \boldsymbol{Q} = (x_3, y_3)$ belongs also to the curve. Generally on elliptic curves, the operation $\boldsymbol{P} + \boldsymbol{P}$, called doubling, has different complexity compared to the addition $\boldsymbol{P} + \boldsymbol{Q}$ with $\boldsymbol{Q} \neq \boldsymbol{P}$.

In practice, it is advantageous to use Jacobian coordinates in order to avoid inverses in $\mathbb{F}_p$. An affine point (x, y) is represented by a triplet $(X : Y : Z)$ such that $x = X/Z^2$ and $y = Y/Z^3$.

Let $n = \#E(\mathbb{F}_p)$ be the cardinality of the group of points $E(\mathbb{F}_p)$. Hasse's theorem states that n is close to p and bounded by: $(\sqrt{p} - 1)^2 \leq n \leq (\sqrt{p} + 1)^2$.

Given a point $\boldsymbol{P} \in E(\mathbb{F}_p)$ and a scalar $d \in \mathbb{N}^*$, we note $[d]\boldsymbol{P}$ the scalar multiplication of $\boldsymbol{P}$ by d. The scalar multiplication is the fundamental operation in most cryptographic algorithms that use elliptic curve arithmetic. In most protocols, the scalar is considered secret and the point public[4].

In the industry, elliptic curve cryptosystems are generally implemented using elliptic curves from standards such as the NIST FIPS186-2 [29], SEC2 [32] or recently generated curves by Bernstein and Lange [5]. All these curves are specified using both efficiency and security criteria. A classic efficiency criterion consists in choosing a special prime, *i.e.* Generalised Mersenne Numbers (GMN) [34], for the finite field $\mathbb{F}_p$. Those primes are sparse, *i.e.* they contain long patterns of zeros or ones, hence due to Hasse's theorem, the orders of the elliptic curves defined over those fields are also sparse.

2.2 Side-Channel Attacks Background

Side-channel analysis, also referred as *Passive Attacks*, was introduced by Kocher *et al.* in [23, 24]. SCA regroups several different techniques. *Simple Side-Channel Analysis* (SSCA) exploits a single execution trace to recover the secret whereas *Differential Side-Channel Analysis* (DSCA) performs statistical treatment on several (possibly millions) traces.

Elliptic curves implementations have been subject to various side-channel attack paths. The simplest one uses SSCA. The attacker's objective is to distinguish a doubling from an addition operation using a single side-channel trace execution.

The principle of the classical DSCA on elliptic curve consists in guessing bit-per-bit (or w-bit per w-bit) the secret scalar and knowing the input point

[4] The problematic is different in pairing-based cryptography where the scalar is generally public and the point secret. We only consider here classic ECC protocols.

manipulated by the implementation. The attacker then recomputes an intermediate guessed value of the algorithm to validate the right guess with a statistical treatment applied to many side-channel execution traces [24, 7]. A recent classification of attacks has categorized all these statistical attacks as *Vertical Analysis*. Indeed, these techniques combine a single time sample t on many side-channel traces to perform the analysis leading to the recovery of the secret data manipulated at this instant t.

Another class of side-channel attack, the *Horizontal Analysis*, has been presented by Clavier *et al.* [12], inspired by the Big Mac attack from Walter [37]. The technique has been later derived to present horizontal attacks on elliptic curves implementations by Hanley *et al.* [19] and Bauer *et al.* [1].

Correlation Analysis. Let $\mathcal{C}^{(i)}$ with $1 \leq i \leq N$ be a set of N side-channel traces captured from a device processing the targeted computations with input value $X^{(i)}$ whose processing occurs at time sample t with l the number of points acquired at time sample t. We consider $\Theta_0 = \{\mathcal{C}^{(1)}(t), \ldots, \mathcal{C}^{(N)}(t)\}$. We denote $S^{(i)}$ with $1 \leq i \leq N$ a set of N guessed intermediate sensible values based on a power model, which is generally linear in the Hamming weight of the data. Let $f(X^{(i)}, \hat{d})$ be a function of the input values $X^{(i)}$ and (a part of) the targeted guessed secret $\hat{d}$. All l points in the leakage trace are equal to this value $f(X^{(i)}, \hat{d})$ for the time sample t. We then consider $\Theta_1 = \{S^{(1)}, \ldots, S^{(N)}\}$. The objective is to evaluate the dependency between both sets Θ_0 and Θ_1 using the Bravais-Pearson correlation factor $\rho(\Theta_0, \Theta_1)$. The correlation value between both series is equal to 1 when the simulated model perfectly matches the measured power traces. It then indicates that the guess on the secret corresponds to the correct key value handled by the device in the computations.

Collision-Correlation Analysis. Correlation can also be used to determine the dependency between different time samples of the same side-channel trace. It will then allow the attacker to detect internal side-channel collisions at two different time samples t_0 and t_1. In this case, the term *collision-correlation* is used. The correlation is applied between the sets $\Theta_0 = \{\mathcal{C}^{(1)}(t_0), \ldots, \mathcal{C}^{(N)}(t_0)\}$ and $\Theta_1 = \{\mathcal{C}^{(1)}(t_1), \ldots, \mathcal{C}^{(N)}(t_1)\}$ where both sets correspond to points of the same side-channel trace taken at different time sample t_0 and t_1. We can expect a maximum correlation value when the same data is processed in the device at the time samples t_0 and t_1. If the attacker can then find a link between this information and the use of the secret, he can recover some information on the secret's value.

2.3 Side-Channel Resistant Scalar Multiplication

On embedded devices, a scalar multiplication needs to be protected against both *Simple Side-Channel Analysis* (SSCA) and *Differential Side-Channel Analysis* (DSCA). To resist SSCA, an attacker should not be able to distinguish an addition from a doubling operation. The main categories of countermeasures are:

– **Regular multiplication algorithms** – Specific scalar multiplication algorithms have been proposed such that they always compute a regular sequence of elliptic curve operations regardless of the value of the secret bits. The double-and-add-always [14] (see Alg. 1), the Montgomery ladder [26, 21] or Joye's double-add [20] are the most well-known examples of regular algorithms. The recently proposed co-Z scalar algorithms [18] are one of the most efficient regular algorithms for ECC over $\mathbb{F}_p$.
– **Unified addition formulæ** – The same formula is used to compute both an addition and a doubling [35].
– **Atomic block** – The addition and doubling operations can be expressed such that the same sequence of field operations are performed. Propositions on the subject are numerous in the literature [10, 17, 31].

The resistance against DSCA can be achieved by using a combination of the following classic countermeasures:

– **Scalar blinding** [14] – We can add a random multiple of the order n of the group $E(\mathbb{F}_p)$ to the scalar d. This alters the representation of d without changing the output of the scalar multiplication. The blinded scalar d' is defined as $d' = d + r.n$ for a random r.
– **Scalar splitting** [9] – The scalar d can be split into several randomized scalars using different methods. The most efficient one consists in an Euclidean splitting [11] by writing $d' = \lfloor d/r \rfloor .r + (d \bmod r)$ for a random r. The scalar multiplication becomes $[d']\boldsymbol{P} = [d \bmod r]\boldsymbol{P} + [\lfloor d/r \rfloor].([r]\boldsymbol{P})$.
– **Randomized projective points** [14] – An affine point $\boldsymbol{P} = (x, y)$ can be represented in Jacobian coordinates as $(\lambda^2 X : \lambda^3 Y : \lambda Z)$ for any nonzero λ. The representation of a point can be randomized by choosing random values of λ.

Algorithm 1 Double-and-add-always

Input: $d = (d_{k-1}, \ldots, d_0)_2 \in \mathbb{N}$ and $\boldsymbol{P} \in E(\mathbb{F}_q)$
Output: $\boldsymbol{Q} = [d]\boldsymbol{P}$

1: $\boldsymbol{R_0} \leftarrow \boldsymbol{O}$; $\boldsymbol{R_1} \leftarrow \boldsymbol{O}$
2: **for** $j = k - 1$ to 0 **do**
3: $\quad \boldsymbol{R_0} \leftarrow [2]\boldsymbol{R_0}$
4: $\quad b \leftarrow d_j$; $\boldsymbol{R_{1-b}} \leftarrow \boldsymbol{R_0} + \boldsymbol{P}$
5: **end for**
6: **return** $\boldsymbol{R_0}$

The scalar blinding countermeasure [14] has been subject to several discussions in previous publications [11, 33] as the order of the curves defined by the NIST is sparse. As we remind in next paragraphs, this property makes the blinding not fully efficient as several bits of the blinded scalar remain unmasked. Thanks to the combination of the recent collision correlation and new horizontal

side-channel attack techniques, we define a new side-channel attack path which takes advantage of this sparse order to complete the full secret exponent recovery. The rest of the paper will consider an implementation using the double-and-add-always (see Alg. 1) in combination with first the scalar blinding technique and then the added randomized projective point countermeasure. Our attacks are applicable to other classical regular algorithm with minor changes as explained in the extended version of this paper [16].

3 Attack on a Blinded Regular Scalar Multiplication with Known Input Point

We first analyze a simple scenario where the input point of the scalar multiplication is known, *i.e.* no DSCA countermeasure on $\boldsymbol{P}$ is used. We consider that the scalar is protected against DSCA using the scalar blinding method. The targeted operation is then $[d']\boldsymbol{P}$ where $d' = d + r.n$ for a random r and n the order of $E(\mathbb{F}_p)$.

Let $\{\mathcal{C}^{(1)}, \ldots, \mathcal{C}^{(N)}\}$ be the N side-channel leakage traces corresponding to the computations $[d'^{(i)}]\boldsymbol{P}^{(i)}$ such that $d'^{(i)} = d + r^{(i)}.n$ are the blinded scalars using random values $r^{(i)}$ and known points $\boldsymbol{P}^{(i)}$ with $1 \leq i \leq N$. We consider that the random factors $r^{(i)}$ are chosen relatively small such that $r^{(i)} \in [0, 2^m - 1]$ with $m \leq 32$ which is the case in many implementations for efficiency reasons.

We first detail the particular form of blinded scalars on standardized curves. Then, we present our attack which is composed of three steps. In a first step, we find the non-masked part of the secret d. Then, we recover each random value $r^{(i)}$ used for the scalar blinding. Finally, we look for the remaining least significant bits of d.

3.1 Representation of the Blinded Scalar using a Sparse Group Order

As noted before, most elliptic curve implementations use in practice curves from public standards [29, 32, 5]. Most standards consider the use of generalised Mersenne numbers to define the prime fields underlying the elliptic curves. These particular primes are very advantageous efficiency-wise as tricks can be applied to improve greatly the modular operations [8].

Classification of sparse group orders. The main standard that defines elliptic curves is the NIST FIPS186-2 [29]. It specifies curves defined over the following primes: $p_{192} = 2^{192} - 2^{64} - 1$, $p_{224} = 2^{224} - 2^{96} + 1$, $p_{256} = 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$, $p_{384} = 2^{384} - 2^{128} - 2^{96} + 2^{32} - 1$ and $p_{521} = 2^{521} - 1$. Due to Hasse's theorem, the orders of the curves defined over each of these fields have also a sparse representation in its upper half. We can categorize them in 3 sets:

- Type-1: the order has a large pattern of ones,
- Type-2: the order has a large pattern of zeros,

– Type-3: the order has a combination of large patterns of both ones and zeros.

Consider the notation $1^{[a,b]}$ with $a, b \in \mathbb{N}$ and $a > b$ a pattern of 1 bits from the bit positions a to b. Similarly, we note $0^{[a,b]}$ a pattern of 0 bits.

Let n, the order of the curve, be a k-bit integer. We can write it depending on its type:

– Type-1: $n = 1^{[k-1,a]} + x$ with $(k-1) > a$ and $0 \leq x < 2^a$,
– Type-2: $n = 2^{k-1} + 0^{[k-2,a]} + x$ with $(k-2) > a$ and $0 \leq x < 2^a$,
– Type-3: $n = 1^{[k-1,a]} + 0^{[a-1,b]} + 1^{[b-1,c]} + x$ with $(k-1) > a > b > c$ and $0 \leq x < 2^c$,

where $a, b, c \in \mathbb{N}$.

Example 1. Here are some standard curves that belong to different types:

– Type-1: $n = 1^{[191,96]} + x$ (NIST P-192 [29]),
– Type-2: $n = 2^{225} + 0^{[224,114]} + x$ (SECP224k1 [32]),
– Type-3: $n = 1^{[255,224]} + 0^{[223,192]} + 1^{[191,128]} + x$ (NIST P-256 [29]).

Form of a random multiple of the order. Let $r \in [1, 2^m - 1]$ be an m-bit random used to mask the secret scalar d such as $d' = d + r.n$. Given the form of the orders of standard curves as seen previously, the mask $r.n$ also has a specific representation.

Let $\tilde{r} = r.(2^m - 1)$ be a $2m$-bit integer, we note $\tilde{r}_1$ and $\tilde{r}_0$ respectively the quotient and remainder of the Euclidean division of $\tilde{r}$ by 2^m. This product has a special form, $\forall r \in [1, 2^m - 1]$ we have:

$$\begin{aligned} \tilde{r}_1 &= r - 1, \\ \tilde{r}_1 + \tilde{r}_0 &= 2^m - 1. \end{aligned}$$

This can be explained by noting the product: $r.(2^m - 1) = (r.2^m) - r$. Hence, $r.2^m$ equals to r followed by m zeros to which we subtract r. This subtraction is performed by computing $2^m - r$ and setting a carry for the most significant part. Hence the higher part equals to $r - 1$. If we add the two halves of $\tilde{r}$, we obtain $(2^m - r) + (r - 1) = 2^m - 1$.

Depending on the category of n, we have the following representations of the mask $r.n$:

– Type-1: $r.n = \tilde{r}_1.2^k + 1^{[k-1,a+m]} + x$, with $0 \leq x < 2^{a+m}$,
– Type-2: $r.n = r.2^k + 0^{[k-1,a+m]} + x$, with $0 \leq x < 2^{a+m}$,
– Type-3: $r.n = \tilde{r}_1.2^k + 1^{[k-1,a+m]} + \tilde{r}_0.2^{a+m} + 0^{[a-1+m,b+m]} + \tilde{r}_1.2^{b+m} + 1^{[b-1+m,c+m]} + x$, with $0 \leq x < 2^{c+m}$.

The patterns of zeros and ones are reduced by m bits for the 3 categories of group orders. Note that these representations of $r.n$ are exact up to possible carries that can happen after each pattern. However, their effect is very limited and does not impact our results.

Adding the random mask r.n to the scalar. The last part of the scalar blinding consists in adding the secret scalar d to the mask $r.n$. First, we observe that an addition $x + (2^m - 1)$ with $x \in [1, 2^m - 1]$ equals to $x - 1$ on the least significant m bits of the results with the $(m+1)$-th bit set at 1.

The notation $d^{[a,b]}$ corresponds to the bits of the scalar d from the bit position a to b. The 3 types of masking representations have an important impact on the (non-)masking of the secret:

- Type-1: $d' = (\tilde{r}_1 + 1).2^k + d^{[k-1,a+m]} + x$, with $0 \leq x < 2^{a+m}$,
- Type-2: $d' = r.2^k + d^{[k-1,a+m]} + x$, with $0 \leq x < 2^{a+m}$,
- Type-3: $d' = (\tilde{r}_1 + 1).2^k + d^{[k-1,a+m]} + \tilde{r}_0.2^{a+m} + d^{[a-1+m,b+m]} + (\tilde{r}_1 + 1).2^{b+m} + d^{[b-1+m,c+m]} + x$, with $0 \leq x < 2^{c+m}$.

Note that the addition of d to $r.n$ can add a carry to the least significant bit of the non masked part of d'.

3.2 First Step: Find the Non-Masked Part of d

From the previous observations on the representation of the blinded scalars $d'^{(i)}$, we can directly deduce chunks of the secret d. We note $\bar{d} = d^{[a,b]}$ the non-masked value of d, for some a, b. We note $\delta = (a-b)$ the bit size of $\bar{d} = (\bar{d}_{\delta-1}, \ldots, \bar{d}_1, \bar{d}_0)_2$. As we do not know the most significant part of the $d'^{(i)}$, we cannot compute an intermediate value based on a guess, we need to perform a *vertical collision-correlation attack*.

For each bit $\bar{d}_j$ of the scalar, a point doubling followed by a point addition are performed where the addition is dummy if $\bar{d}_j = 0$. If $\bar{d}_j = 1$, all the results of point doubling and point addition are used whereas, if $\bar{d}_j = 0$, the result of the point addition is discarded. This means that the next point doubling will take the same input as the previous point addition when $\bar{d}_j = 0$, resulting in a collision. We use the notations In, respectively Out, to indicate the input, respectively output, of a given operation.

1. To find the j-th bit $\bar{d}_j$ of $\bar{d}$ with $0 < j < \delta$, identify the two elliptic curve operations that possibly correspond to its processing. The processing of a bit $\bar{d}_j = 0$ generates a collision between the input of the point addition $\mathsf{ECADD}(j)$ and the input of the next point doubling $\mathsf{ECDBL}(j+1)$ whereas there is no collision when $\bar{d}_j = 1$.
2. Construct a first vector $\Theta_0 = \left\{\mathcal{C}^{(i)}(t_0)\right\}_{1 \leq i \leq N}$ that corresponds to the time sample t_0 of the N leakage traces $\mathcal{C}^{(i)}$. The instant t_0 corresponds to the computation of $\mathsf{In}(\mathsf{ECADD}(j))$.
3. Construct similarly a second vector $\Theta_1 = \left\{\mathcal{C}^{(i)}(t_1)\right\}_{1 \leq i \leq N}$ that corresponds to the time sample t_1 of the N leakage traces $\mathcal{C}^{(i)}$. The instant t_1 corresponds to the computation of $\mathsf{In}(\mathsf{ECDBL}(j+1))$.
4. Perform a collision-correlation analysis $\rho(\Theta_0, \Theta_1)$. We can expect that the correlation coefficient will be maximal when the operations $\mathsf{ECADD}(j)$ and $\mathsf{ECDBL}(j+1)$ take the same input point, hence when $\bar{d}_j = 0$.

Remark 1. Note that, for the Type-3 orders, the attack has to be repeated on each interval of non-masked bits of d.

Remark 2. We remind that the success rate of collision-correlation attacks can heavily depend on the choice of the threshold value. A discussion of this point based on practical results is given in Section 5.1.

3.3 Second Step: Retrieve Random Masks with Horizontal Attacks

From Section 3.1, we know that the random r used in the scalar blinding directly appears in the most significant part of d'. The second part of our attack consists in retrieving the random values $r^{(i)} \in [1, 2^m - 1]$ from each blinded scalar $d'^{(i)}$ using an *horizontal correlation attack*. The following attack procedure is repeated for each trace $\mathcal{C}^{(i)}$, $1 \leq i \leq N$:

1. Try all possible m-bit values of $r^{(i)}$. In most implementations the random chosen for the scalar blinding is small, *i.e.* $r \leq 2^{32}$, hence this enumeration is generally feasible. A guess on $r^{(i)}$ directly gives a guess on the first m bits[5] of $d'^{(i)}$.
2. Let $\hat{r}$ be the guess on $r^{(i)}$. This guess gives the attacker a sequence of elliptic curve operations that appear at the beginning of the trace $\mathcal{C}^{(i)}$. Since the attacker knows the input point $\boldsymbol{P}^{(i)}$, he can compute the sequence of multiples of $\boldsymbol{P}^{(i)}$ that should be processed for a given $\hat{r}$. Note that from the previous section, we also know the following δ bits of the non-masked part of the blinded scalar. Then η intermediate points can be computed with[6] $\eta = 2(m + \delta)$.
3. Choose a leakage model function $\mathcal{L}$, *e.g.* the Hamming weight, and compute some predicted values derived from the η points T_j, $1 \leq j \leq \eta$. The attacker computes the values $l_j = \mathcal{L}(T_j)$ for $1 \leq j \leq \eta$ and creates the vector $\Theta_1 = (l_j)_{1 \leq j \leq \eta}$.
4. Construct η sub-traces from the trace $\mathcal{C}^{(i)}$ where the targeted values T_j, $1 \leq j \leq \eta$ are manipulated. The attacker constructs the vector $\Theta_0 = (o_j)_{1 \leq j \leq \eta}$ where o_j are the identified points of interest related to T_j.
5. Compute the correlation coefficient $\rho(\Theta_0, \Theta_1)$. If the guess $\hat{r}$ is correct, the sequence of T_j is also correct, hence we can expect a maximal coefficient of correlation.

Remark 3. The random r appears at the beginning of each pattern of ones in the order n. Hence, on curves of Type-3, the attacker could exploit this property to obtain more time samples per trace to recover the random values.

---

[5] Note that $(\tilde{r}_1^{(i)} + 1) = r^{(i)}$ for Type-1 and Type-3 orders.

[6] Depending on the point addition and point doubling formulæ used, an attacker could also include intermediate long-integer operations in order to work with even larger sets.

3.4 Third Step: Recover the Least Significant Part of d

From the previous parts of the attack, we know the most significant part of d as well as the random values $r^{(i)}$ of each blinded scalar $d'^{(i)}$. We need to recover the least significant part of the secret. By guessing the next w unknown bits of d, we can compute guessed blinded scalars $\hat{d}'^{(i)}$. We can then perform a classical *vertical correlation attack* to validate the guesses. The following steps need to be repeated until d is fully recovered (directly or with an easy brute-force):

1. Guess the following w unknown bits of d. From this guess and the known random $r^{(i)}$, compute the N guessed blinded scalars $\hat{d}'^{(i)}$ for $1 \leq i \leq N$.
2. Choose a leakage model function $\mathcal{L}$. For the i-th curve, the attacker can compute some predicted values derived from the η points $T_j^{(i)}$, $1 \leq j \leq \eta$ with $\eta = 2w$. He creates the vector $\Theta_1 = \left(l_j^{(i)}\right)_{i,j}$, with $1 \leq j \leq \eta$, $1 \leq i \leq N$ and where $l_j^{(i)} = \mathcal{L}\left(T_j^{(i)}\right)$.
3. Construct a vector $\Theta_0 = \left(o_j^{(i)}\right)_{i,j}$ where $o_j^{(i)}$ is the point of interest of the trace $\mathcal{C}^{(i)}$ corresponding to the processing of $T_j^{(i)}$.
4. Compute the correlation coefficient $\rho(\Theta_0, \Theta_1)$. We can expect a maximal correlation coefficient when the w guessed bits are correct, hence the η intermediate points of the N traces are correct.

Remark 4. Note that there can be a carry on the least significant bit of the w guessed bits of $\hat{d}'^{(i)}$. If a wrong guess is recovered in first position due to the carry, the following attack on the next w bits will give low correlation values. The attacker then needs to correct the previous guess with a carry in order to continue his attack.

4 Attack on a Protected Scalar Multiplication

The main attack strategy proposed in the previous section can also be applied on an implementation with point blinding. The first step is identical even with unknown input points. However as the input is unknown, classical correlation attacks where a guessed intermediate variable is correlated to leakage observations are not applicable anymore. We present in this section modifications to the second and third steps of our previous attack to recover the full secret scalar on a fully protected scalar multiplication.

4.1 First Step: Vertical Collision-Correlation

The first attack is identical to the known input point scenario. The proposed vertical collision-correlation in Section 3.2 does not require the knowledge of the inputs. Hence the same steps can be applied in the unknown input case in order to recover the non-masked bits of the scalar d, *i.e.* $\bar{d}$ of bit length δ.

4.2 Second Step: Horizontal Collision-Correlation

The horizontal correlation attack presented in Section 3.3 is not applicable without a known input point. We need to perform an *horizontal collision-correlation* on each leakage trace $\mathcal{C}^{(i)}$, $1 \leq i \leq N$, simply noted $\mathcal{C}$ below for readability:

1. Try all possible m-bit values of $r^{(i)}$.
2. The guessed random $\hat{r}$ gives the attacker the supposed starting sequence of elliptic curve operations that appears in the scalar multiplication. The known part of d also provides the following δ bits of the blinded scalar. Hence, the attacker works with $(m+\delta)$ bits of the blinded scalar $\hat{d}'$. The processing of a bit at 0 or 1 generates different possible collisions between elliptic curve coordinates:
   - if $\hat{d}'_j = 1$, we have a collision between the coordinates of the output of $\mathsf{ECADD}(j)$ and the coordinates of the input point of $\mathsf{ECDBL}(j+1)$,
   - if $\hat{d}'_j = 0$, we have a collision between the coordinates of the input of $\mathsf{ECADD}(j)$ and the coordinates of the input of $\mathsf{ECDBL}(j+1)$.
3. Construct two vectors Θ_0 and Θ_1 corresponding to different time samples of the leakage trace $\mathcal{C}$. They are defined as:

$$\Theta_0 = \left\{\mathcal{C}\left(t_0^X(j)\right), \mathcal{C}\left(t_0^Y(j)\right), \mathcal{C}\left(t_0^Z(j)\right)\right\}_{0 \leq j < (m+\delta)},$$
$$\Theta_1 = \left\{\mathcal{C}\left(t_1^X(j)\right), \mathcal{C}\left(t_1^Y(j)\right), \mathcal{C}\left(t_1^Z(j)\right)\right\}_{0 \leq j < (m+\delta)},$$

   where

$$t_0^X(j) = \begin{cases} \mathsf{Out}^X\left(\mathsf{ECADD}(j)\right) & \text{if } \hat{d}'_j = 1, \\ \mathsf{In}^X\left(\mathsf{ECADD}(j)\right) & \text{if } \hat{d}'_j = 0, \end{cases}$$
$$t_1^X(j) = \mathsf{In}^X\left(\mathsf{ECDBL}(j+1)\right),$$

   respectively t_0^Y, t_1^Y and t_0^Z, t_1^Z for the Y and Z coordinates of the corresponding elliptic points. The notations In and Out represent the time samples of the processing of respectively the input point and output point coordinates of the parametrized elliptic curve operation.
4. Compute the correlation analysis $\rho(\Theta_0, \Theta_1)$. For the correct guess $\hat{r}$, the sequence of collisions is correct and should give the maximum coefficient of correlation.

4.3 Third Step: Vertical Collision-Correlation

We need to apply a *vertical collision-correlation* side-channel attack in this third step as the input is unknown. Instead of recomputing the intermediate points of the scalar multiplication corresponding to guesses on d and computing a correlation with the leakage observation, we build collision vectors, as previously, depending on the bit values of the guess:

1. Guess w unknown bits of d. From this guess and the known random $r^{(i)}$, we can compute guessed blinded scalars $\hat{d}'^{(i)}$ for $1 \leq i \leq N$.

2. Construct collision vectors Θ_0 and Θ_1 as defined in the previous attack depending on the values of the bits of $\hat{d}'^{(i)}$. If we consider that $u \leq \delta$ bits of d are already recovered, the collision vectors are of size $(m + u + w)N$.
3. Compute the correlation analysis $\rho(\Theta_0, \Theta_1)$. For the correct w guessed bits, we can expect the highest correlation coefficient.

Remark 5. In order to find the bit d_j, the collision should be evaluated on the operations of the next iteration $(j + 1)$ of the scalar multiplication. Hence, the final least significant bit cannot be recovered using the attack but has to be guessed.

5 Experimentations

In order to validate our different attack paths on the blinded scalar multiplication, we performed simulations on a double-and-add-always algorithm using the standardized elliptic curve P-192 from NIST. For our implementation, we chose the classical jacobian projective coordinates and used the most efficient generic addition and doubling algorithms[7]. The particular choice of coordinates or group operation algorithms has no impact on the feasibility of our attacks. Its only effect is on the selection of time samples on which to compute correlations or collisions. We performed our attacks using 8-bit and 16-bit random for the scalar blinding. As the use of larger random size impacts the computational time of the attacks, we chose small random sizes in order to repeat several hundred of times our attacks for consistency.

Our simulation traces consist of the leakage of the inputs and outputs of long integer operations (multiplication, squaring, addition) that are used for the elliptic curve group operations. The leakage is modeled with the classical Hamming weight function. As nowadays most arithmetic coprocessors and chip have 32-bit architectures, we consider Hamming weight leakage of words of 32-bits[8]. Hence, the leakage of the long-integer multiplication $c = a.b \bmod p$ is represented by the vector $(\mathsf{HW}_{32}(a_i), \mathsf{HW}_{32}(b_i), \mathsf{HW}_{32}(c_i))$ where $\mathsf{HW}_{32}(a_i)$, respectively $\mathsf{HW}_{32}(b_i)$ and $\mathsf{HW}_{32}(c_i)$, represents the Hamming weight of the i-th 32-bit word of a, respectively b and c. We performed our simulations with different level of noise having a Gaussian distribution with mean 0 and standard deviation σ. Finally, we use the Pearson correlation as side-channel distinguisher.

5.1 Simulated Attack Results on Known Input Points

We first present results on the attack path with a known (non-masked) input point from Section 3. Table 1 details the success rates obtained for the three

[7] We selected the addition algorithm *add-2007-bl* with complexity $11\mathsf{M} + 5\mathsf{S}$ and the doubling algorithm *dbl-2007-bl* with complexity $1\mathsf{M} + 8\mathsf{S}$ from [3].

[8] We expect the horizontal parts of our attacks to give better results on smaller architectures as more time samples will be available per long integer number.

attack steps with various parameters. We recall that the parameter N is the number of traces and m is the bit size of the random for the exponent blinding.

The first step of the attack is a vertical collision-correlation. We tested its success using 500 and 1000 leakage traces. The results show a great success rate even when the noise becomes quite high. We can expect even better success rate for high σ if the attacker has access to more traces. Figure 1 illustrates the spreading of the correlation coefficient around its mean value. We clearly see the variance of the coefficient increasing for high levels of noise when a collision happens, *i.e.* the bit equals 0. This figure also gives a good idea on the threshold value for the correlation coefficient in practice, in order to decide if a collision happened. Its selection needs to be more precise the higher the noise level to obtain a good success rate. In practice, we observe that the last bits found by the attack are sometimes different to the expected scalar d. This is due to a possible carry propagation because of the addition of the masking value $r.n$. In this case, a bit equal to 1 is found as the correlation coefficient becomes low. This possible error is then corrected during the third part of the attack where the attacker can start the analysis a few bits before the ones retrieved at this step.

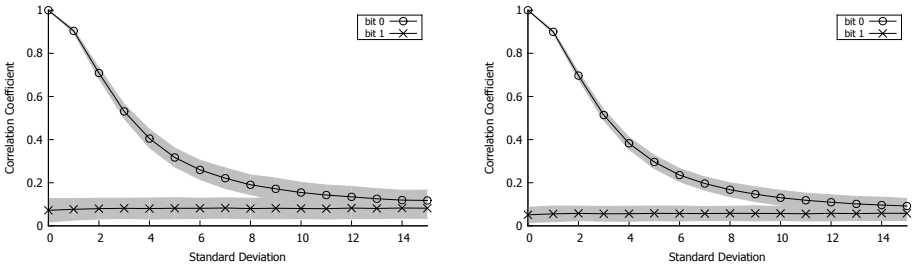

Fig. 1. First attack step: correlation coefficient spreading, left for 500 traces, right for 1000 traces.

The second attack step is an horizontal correlation that needs to be repeated for each trace. As the horizontal attack uses only one trace, the parameters affecting its success rate are the size m of the random used for the exponent blinding as well as the noise level σ. A larger random gives more time samples per trace, hence better results for our attack. However, as we enumerate 2^m values, the computational times may be prohibitive for large bit sizes of random. The attack also uses the bits recovered in the first step to compute guessed intermediate variables and perform a correlation on even more time samples. The success rates are then very good even in the presence of high noise.

The last attack step is a vertical correlation. As the first part, we performed tests on 500 and 1000 traces to compare the evolution of the success rate. The results are very good until strong levels of noise ($\sigma > 10$).

Remark 6. As explained in Section 3.4, due to possible carry propagation instead of recovering the right guess we can obtain the correct guess ± 1. However, we will be immediately informed as the correlation coefficients for the attack on the next w bits will be much lower. We consider the attack successful if the best guess is close to the right guess (± 1).

Attack steps	N	m	**Standard Deviation σ**					
			0	**1**	**2**	**5**	**10**	**15**
Vertical	500	-	1.0	1.0	1.0	1.0	0.88	0.74
collision-correlation	1000	-	1.0	1.0	1.0	1.0	0.99	0.76
Horizontal	-	8	1.0	1.0	1.0	1.0	1.0	0.77
correlation	-	16	1.0	1.0	1.0	1.0	1.0	0.85
Vertical	500	-	1.0	1.0	1.0	1.0	0.64	0.42
correlation	1000	-	1.0	1.0	1.0	1.0	0.84	0.52

Table 1. Success rate for known input points.

5.2 Simulated Attack Results on Unknown Input Points

We now present results on the attack paths from Section 4 on a fully protected scalar multiplication with scalar blinding and point randomization. Table 2 presents the success rates of the second and third steps as the first vertical collision-correlation is identical. Hence, the results from Figure 1 and the first row of Table 1 also apply to the unknown input point case.

The second step is an horizontal collision-correlation attack. Its success rate depends on the number of time samples considered in each trace. The same problematic as in the known-point case is present, *i.e.* a larger random gives better results for a higher computational cost. The success rate drops quicker than previous attacks for higher levels of noise. Indeed, the attack only uses time samples of computations on coordinates of intermediate elliptic curve points. Hence, contrary to vertical attacks the attacker is limited to a fixed number of time samples regardless of the noise level.

The third attack step is a vertical collision-correlation. As each vertical attack, we tested its success rate on 500 and 1000 traces. Its efficiency is very high even with a strong noise. The Remark 6 also applies here as possible carries can appear.

From our simulations, we observe that in the unknown input point case our attack retrieves the full scalar for noise levels up to $\sigma \approx 5$ whereas our attack works up to $\sigma \approx 10$ with a known input point.

Attack steps	N	m	Standard Deviation σ					
			0	**1**	**2**	**5**	**10**	**15**
Horizontal	-	8	1.0	1.0	0.9	0.1	0.02	0.01
collision-correlation	-	16	1.0	1.0	0.95	0.23	0.10	0.02
Vertical	500	-	1.0	1.0	1.0	1.0	1.0	0.97
collision-correlation	1000	-	1.0	1.0	1.0	1.0	1.0	0.99

Table 2. Success rate for unknown input points.

6 Countermeasures

We propose here countermeasures that could be applied at different levels of the implementation.

Scalar splitting. The Euclidean splitting proposed in [11]: $[d]\boldsymbol{P} = [d \bmod r]\boldsymbol{P} + [\lfloor d/r \rfloor]([r]\boldsymbol{P})$ is generally preferred to the additive splitting [9] that could be vulnerable to advanced attacks [28] and to the multiplicative splitting [36] that requires a costly modular inversion. However the Euclidean splitting still remains less efficient than the scalar blinding and can be disregarded by developers. Note that exponent splitting with a mask of bit length m could be surmounted with $2^{m/2}$ traces due to the birthday paradox. The use of a scalar splitting method, with large enough random masks, thwarts the proposed attacks on standard curves.

Scalar blinding with larger random. As our attack path exploits the fact that, for small random values, the scalar blinding countermeasure does not mask part of the scalar, a possible solution could be to use larger random. A first selection parameter for the random size could be to have an implementation where all scalar bits are masked for the supported elliptic curves. Let $\mathcal{P}$ be the largest pattern size amongst all curves' order that are supported by an application[9]. Hence, in order to use the scalar blinding countermeasure, one would need to implement a random size m such that $m > \mathcal{P}$ to obtain a scalar fully masked.

A second selection parameter could be to select a random size large enough such that our proposed attack path is no more applicable. Indeed, in our second attack step the attacker needs to try all possible random values. Let $\mathcal{B}$ be the maximum brute force capability of an attacker, *i.e.* he can perform $2^{\mathcal{B}}$ operations in reasonable time. Hence, one would need to choose a random size m such that $m > \mathcal{B}$.

Generally, the more restrictive selection criterion is $m > \mathcal{B}$, as $\mathcal{B} << \mathcal{P}$ for most standardized curves. The overhead added to the scalar multiplication complexity by the larger m value then needs to be compared to other countermeasures. For example, the scalar splitting that has an overhead factor of 1.5 which can be more advantageous depending on the implementation requirements.

[9] For example, amongst all NIST curves, the P-521 has the largest pattern of ones in its order with a pattern size of 262 bits, *i.e.* $\mathcal{P} = 262$.

Atomic algorithm and unified formulæ. Our attack only targets regular scalar multiplication algorithms, hence an atomic algorithm could be considered. There are many atomic formulas for elliptic curves proposed in the literature [10, 17, 31]. This countermeasure generally offers an interesting time/memory trade-off for embedded devices. However a recent attack was presented by Bauer *et al.* [1] against the main atomic formulæ. Even if the practicality of their attack is subject to different parameters, it clearly demonstrates a vulnerability in many atomic schemes. As mentioned by the authors of [1], their technique can also be applied to unified formulas on Weierstraß curves [6] as well as Edward's curves [4].

7 Conclusion

We present in this paper a new side-channel attack combination targeting elliptic curves implementations of regular scalar multiplication on some standardized curves. We assume the scalar multiplication algorithm implements the classical scalar blinding and point randomization techniques, two of the most used countermeasures against differential side-channel attacks. Our attacks exploit the previously known weakness of the sparse order of the standardized curves in order to fully recover a blinded secret scalar. We combine techniques from collision correlation analysis as well as horizontal and vertical attack to design a new attack path.

Acknowledgments. The authors would like to thank Vincent Verneuil for his detailed and perceptive comments.

References

1. Bauer, A., Jaulmes, E., Prouff, E., Wild, J.: Horizontal collision correlation attack on elliptic curves. In: Selected Areas in Cryptography (2013)
2. Bauer, A., Jaulmes, É.: Correlation analysis against protected SFM implementations of RSA. In: Paul, G., Vaudenay, S. (eds.) Progress in Cryptology - INDOCRYPT 2013 - 14th International Conference on Cryptology in India, Mumbai, India, December 7-10, 2013. Proceedings, Lecture Notes in Computer Science, vol. 8250, pp. 98–115. Springer (2013)
3. Bernstein, D.J., Lange, T.: Explicit-formulas database. http://hyperelliptic.org/EFD/g1p/auto-shortw.html
4. Bernstein, D.J., Lange, T.: Faster addition and doubling on elliptic curves. In: Kurosawa, K. (ed.) Advances in Cryptology – ASIACRYPT 2007, Lecture Notes in Computer Science, vol. 4833, pp. 29–50. Springer Berlin Heidelberg (2007)
5. Bernstein, D.J., Lange, T.: Safecurves: choosing safe curves for elliptic-curve cryptography (accessed 26 May 2014), http://safecurves.cr.yp.to
6. Brier, E., Joye, M.: Weierstraß elliptic curves and side-channel attacks. In: Naccache, D., Paillier, P. (eds.) Public Key Cryptography, Lecture Notes in Computer Science, vol. 2274, pp. 335–345. Springer Berlin Heidelberg (2002)

7. Brier, E., Clavier, C., Olivier, F.: Correlation power analysis with a leakage model. In: Joye, M., Quisquater, J.J. (eds.) Cryptographic Hardware and Embedded Systems - CHES 2004, Lecture Notes in Computer Science, vol. 3156, pp. 135–152. Springer Berlin / Heidelberg (2004)
8. Brown, M., Hankerson, D., López, J., Menezes, A.: Software implementation of the NIST elliptic curves over prime fields. In: Naccache, D. (ed.) Topics in Cryptology - CT-RSA 2001, Lecture Notes in Computer Science, vol. 2020, pp. 250–265. Springer Berlin Heidelberg (2001)
9. Chari, S., Jutla, C.S., Rao, J.R., Rohatgi, P.: Towards sound approaches to counteract power-analysis attacks. In: Wiener, M. (ed.) Advances in Cryptology – CRYPTO'99, Lecture Notes in Computer Science, vol. 1666, pp. 398–412. Springer Berlin Heidelberg (1999)
10. Chevallier-Mames, B., Ciet, M., Joye, M.: Low-cost solutions for preventing simple side-channel analysis: Side-channel atomicity. IEEE Transactions on Computers 53, 760–768 (2004)
11. Ciet, M., Joye, M.: (Virtually) free randomization techniques for elliptic curve cryptography. In: Qing, S., Gollmann, D., Zhou, J. (eds.) Information and Communications Security, Lecture Notes in Computer Science, vol. 2836, pp. 348–359. Springer Berlin Heidelberg (2003)
12. Clavier, C., Feix, B., Gagnerot, G., Roussellet, M., Verneuil, V.: Horizontal correlation analysis on exponentiation. In: Information and Communications Security, Lecture Notes in Computer Science, vol. 6476, pp. 46–61. Springer Berlin Heidelberg (2010)
13. Clavier, C., Feix, B., Gagnerot, G., Giraud, C., Roussellet, M., Verneuil, V.: ROSETTA for single trace analysis. In: Galbraith, S., Nandi, M. (eds.) Progress in Cryptology - INDOCRYPT 2012, Lecture Notes in Computer Science, vol. 7668, pp. 140–155. Springer Berlin Heidelberg (2012)
14. Coron, J.S.: Resistance against differential power analysis for elliptic curve cryptosystems. In: Koç, Ç.K., Paar, C. (eds.) Cryptographic Hardware and Embedded Systems - CHES 1999. Lecture Notes in Computer Science, vol. 1717, pp. 292–302. Springer Berlin / Heidelberg (1999)
15. Diffie, W., Hellman, M.E.: New directions in cryptography. IEEE Transactions on Information Theory 22(6), 644–654 (1976)
16. Feix, B., Roussellet, M., Venelli, A.: Side-channel analysis on blinded regular scalar multiplications. IACR Cryptology ePrint Archive (2014)
17. Giraud, C., Verneuil, V.: Atomicity improvement for elliptic curve scalar multiplication. In: Gollmann, D., Lanet, J.L., Iguchi-Cartigny, J. (eds.) Smart Card Research and Advanced Application, Lecture Notes in Computer Science, vol. 6035, pp. 80–101. Springer Berlin Heidelberg (2010)
18. Goundar, R., Joye, M., Miyaji, A., Rivain, M., Venelli, A.: Scalar multiplication on Weierstraß elliptic curves from co-z arithmetic. Journal of Cryptographic Engineering 1(2), 161–176 (2011)
19. Hanley, N., Kim, H., Tunstall, M.: Exploiting collisions in addition chain-based exponentiation algorithms. Cryptology ePrint Archive, Report 2012/485 (2012)
20. Joye, M.: Highly regular right-to-left algorithms for scalar multiplication. In: Paillier, P., Verbauwhede, I. (eds.) Cryptographic Hardware and Embedded Systems - CHES 2007, Lecture Notes in Computer Science, vol. 4727, pp. 135–147. Springer Berlin Heidelberg (2007)
21. Joye, M., Yen, S.M.: The Montgomery powering ladder. In: Kaliski, B., Ko, e., Paar, C. (eds.) Cryptographic Hardware and Embedded Systems - CHES 2002, Lecture

Notes in Computer Science, vol. 2523, pp. 291–302. Springer Berlin Heidelberg (2003)
22. Koblitz, N.: Elliptic curve cryptosystems. Mathematics of computation 48, 203–209 (1987)
23. Kocher, P.: Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In: Koblitz, N. (ed.) Advances in Cryptology – CRYPTO '96, Lecture Notes in Computer Science, vol. 1109, pp. 104–113. Springer Berlin / Heidelberg (1996)
24. Kocher, P., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M. (ed.) Advances in Cryptology - CRYPTO' 99, Lecture Notes in Computer Science, vol. 1666, pp. 789–789. Springer Berlin / Heidelberg (1999)
25. Miller, V.: Use of elliptic curves in cryptography. In: Advances in Cryptology – CRYPTO'85 Proceedings. pp. 417–426 (1986)
26. Montgomery, P.L.: Speeding the Pollard and elliptic curve methods of factorization. Mathematics of computation 48(177), 243–264 (1987)
27. Moradi, A., Mischke, O., Eisenbarth, T.: Correlation-enhanced power analysis collision attack. In: Mangard, S., Standaert, F.X. (eds.) CHES. Lecture Notes in Computer Science, vol. 6225, pp. 125–139. Springer (2010)
28. Muller, F., Valette, F.: High-order attacks against the exponent splitting protection. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T. (eds.) Public Key Cryptography - PKC 2006, Lecture Notes in Computer Science, vol. 3958, pp. 315–329. Springer Berlin Heidelberg (2006)
29. National Institute Standards and Technology: Digital Signature Standard (DSS). Publication 186–2 (2000)
30. Rivest, R., Shamir, A., Adleman, L.: A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM 21, 120–126 (1978)
31. Rondepierre, F.: Revisiting atomic patterns for scalar multiplications on elliptic curves. In: 12th Smart Card Research and Advanced Application Conference (2013)
32. SEC2: Standards for Efficient Cryptography Group/Certicom Research. Recommanded Elliptic Curve Cryptography Domain Parameters (2000)
33. Smart, N., Oswald, E., Page, D.: Randomised representations. IET Information Security 2, 19–27(8) (June 2008)
34. Solinas, J.: Generalized Mersenne numbers. Technical report CORR-39, Dept. of C&O, University of Waterloo (1999)
35. Stebila, D., Thériault, N.: Unified point addition formulæ and side-channel attacks. In: Goubin, L., Matsui, M. (eds.) Cryptographic Hardware and Embedded Systems - CHES 2006, Lecture Notes in Computer Science, vol. 4249, pp. 354–368. Springer Berlin Heidelberg (2006)
36. Trichina, E., Bellezza, A.: Implementation of elliptic curve cryptography with built-in counter measures against side channel attacks. In: Cryptographic Hardware and Embedded Systems - CHES 2002, Lecture Notes in Computer Science, vol. 2523, pp. 98–113. Springer Berlin Heidelberg (2003)
37. Walter, C.: Sliding windows succumbs to Big Mac attack. In: Cryptographic Hardware and Embedded Systems – CHES 2001, Lecture Notes in Computer Science, vol. 2162, pp. 286–299. Springer Berlin Heidelberg (2001)
38. Witteman, M., van Woudenberg, J., Menarini, F.: Defeating RSA multiply-always and message blinding countermeasures. In: Kiayias, A. (ed.) Topics in Cryptology – CT-RSA 2011, Lecture Notes in Computer Science, vol. 6558, pp. 77–88. Springer Berlin / Heidelberg (2011)