To Infinity and Beyond: Combined Attack on ECC using Points of Low Order[⋆]

Junfeng Fan, Benedikt Gierlichs, and Frederik Vercauteren

Katholieke Universiteit Leuven, COSIC & IBBT
Kasteelpark Arenberg 10, B-3001 Leuven-Heverlee, Belgium
`firstname.lastname@esat.kuleuven.be`

Abstract. We present a novel combined attack against ECC implementations that exploits specially crafted, but valid input points. The core idea is that after fault injection, these points turn into points of very low order. Using side channel information we deduce when the point at infinity occurs during the scalar multiplication, which leaks information about the secret key. In the best case, our attack breaks a simple and differential side channel analysis resistant implementation with input/output point validity and curve parameter checks using a single query.

Keywords: fault attack, side channel attack, elliptic curve cryptography

1 Introduction

Elliptic curve cryptography (ECC) is a public-key cryptosystem that was independently proposed by Miller [33] and Koblitz [29]. In the context of embedded implementations, ECC is an interesting alternative to systems like RSA [37] because it allows for more compact and more efficient implementations.

The ubiquity of embedded cryptography in applications such as smart cards, RFID tags, access control, etc. leads to a new security threat that does not target the mathematical strength of the cryptographic algorithms but the physical strength of concrete implementations using side channel and fault attacks. Side channel attacks (SCAs) were first described by Kocher in [30] and use the fact that physical devices leak information through measurable quantities such as power consumption [31], timing behavior [30], electromagnetic radiation [24, 36], etc. Fault attacks (FAs) were introduced by Boneh et al. [10], and rely on the fact that an adversary can actively inject faults into a device which typically leads the device to compute an incorrect result. Ways to inject faults include clock and power glitches [4, 6], lasers [38], etc.

[⋆] This work was supported in part by the European Commission's ECRYPT II NoE (ICT-2007-216676), by the Belgian State's IAP program P6/26 BCRYPT, by the K.U. Leuven-BOF (OT/06/40) and by the Research Council K.U. Leuven: GOA TENSE (GOA/11/007). Benedikt Gierlichs and Frederik Vercauteren are Postdoctoral Fellows of the Fund for Scientific Research - Flanders (FWO).

Straightforward implementations of ECC can be easily broken by a range of well known attacks, including simple and differential side channel analysis (SSCA, DSCA) as shown by Coron [20] and differential fault analysis as demonstrated by Biehl et al. [9] and later generalized by Ciet and Joye [16]. We refer to Fan et al. [23] for a comprehensive overview of the existing countermeasures to thwart these attacks and simply focus on the main ideas.

Resistance against SSCA can be achieved by regular scalar multiplication algorithms [20, 28], unified addition and doubling formulae [12, 19] or side channel atomicity [14]. Basically any solution that ensures a constant sequence of operations in the scalar multiplication algorithm, identical or indistinguishable point operations, is viable.

DSCA can be thwarted by ensuring that the scalar multiplication algorithm processes strictly unpredictable, e.g. randomized, data. Typical randomization techniques include base point blinding [20], randomized projective coordinates [20], curve isomorphisms [27] and field isomorphisms [27]. Alternative approaches include key randomization [20] and random key splitting [15] before each scalar multiplication, but they require that an adversary cannot extract any information from a single trace [18]. However, as shown by Goubin [25] most of these countermeasures can be broken in the chosen message scenario when the curve admits "special points", i.e. where one of the coordinates is zero. Smart [40] provides several easy countermeasures preventing Goubin's attack: for special points of low order, cofactor multiplication is proposed and to avoid special points of large order, all points are first mapped to an isogenous curve, before scalar multiplication is executed. Note that all NIST curves over large prime fields have cofactor equal to one.

Due to ECC's group structure, an elegant and efficient way to detect faults is to check if the input to and the output of the scalar multiplication algorithm are valid points on the curve as explained by Biehl et al. [9]. Ciet and Joye point out that one must additionally check the curve parameters for faults [16], which in the remainder of the paper we consider to be part of the initial validity check.

In this paper we present a novel attack that combines fault injection with SSCA (cf. combined attack [3]) and specially crafted, but valid input points P. The core idea is that, after a single fault injection, P turns into a point P' of very low order ℓ (e.g. $\ell = 2, 3, \ldots, 200$) with practical probability. Since the point P' has low order, the point at infinity will appear during the computation of $k \cdot P'$. This event can be detected via side channels and leaks information about the key k. Our attack cannot be prevented by most of the countermeasures mentioned above such as input and output validity checks, cofactor multiplication and isogeny defence (which foil Goubin's attack), SSCA countermeasures and it bypasses many DSCA countermeasures.

The paper is organized as follows. In Section 2, we recall the necessary background on elliptic curves and in Section 3, we describe an effective algorithm to compute valid points on an elliptic curve that, after a bit-flip in one of their coordinates turn into points of a given small order. In Section 4, we exploit these points to derive our new attack and illustrate it on a very basic implementation.

In Section 5, we discuss the assumptions underlying our attack and analyze its applicability when the basic implementation is enhanced with common countermeasures. Finally, Section 6 concludes the paper.

2 Background on elliptic curves

In this section we briefly review the necessary background on elliptic curves over $\mathbb{F}_p$. An elliptic curve E over $\mathbb{F}_p$ with $p > 3$ can always be given by a short Weierstrass equation $y^2 = x^3 + ax + b$, with $a, b \in \mathbb{F}_p$ and $4a^3 + 27b^2 \neq 0$. For every finite field K containing $\mathbb{F}_p$ one now considers the set of K-rational points

$$E(K) := \{(x, y) \in K \times K \mid y^2 = x^3 + ax + b\} \cup \{\mathcal{O}\}$$

where $\mathcal{O}$ denotes the point at infinity.

2.1 Group law

The use of elliptic curves in cryptography stems from the fact that $E(K)$ naturally possesses the structure of an abelian group. It is common practice to denote the group operations in an additive way (i.e. using $+$ and $-$ symbols), as opposed to the multiplicative notation when dealing with groups like $\mathbb{F}_p^*$. The group law is defined by the following general rules: $\mathcal{O}$ is the zero element, and any three points that lie on a line add up to zero.

Group law formulae Working this out yields the following explicit rules for adding two points $P = (x_P, y_P)$ and $Q = (x_Q, y_Q)$. If $Q = -P$, i.e. if $x_P = x_Q$ and $y_P = -y_Q$, then $P + Q = \mathcal{O}$. If $P \neq \pm Q$, we obtain the following addition formula: $R = (x_R, y_R) = P + Q$ with

$$x_R = \left(\frac{y_Q - y_P}{x_Q - x_P}\right)^2 - x_P - x_Q \quad \text{and} \quad y_R = \left(\frac{y_Q - y_P}{x_Q - x_P}\right)(x_P - x_R) - y_P\,. \quad (1)$$

If $P = Q$, we obtain the doubling formula: $R = (x_R, y_R) = 2 \cdot P$ with

$$x_R = \left(\frac{3x_P^2 + a}{2y_P}\right)^2 - x_P - x_Q \quad \text{and} \quad y_R = \left(\frac{3x_P^2 + a}{2y_P}\right)(x_P - x_R) - y_P\,. \quad (2)$$

Note that the above formula for addition does not depend on the curve equation at all and that the formula for doubling only involves the parameter a. This simple fact has been exploited in several attacks before [9] and will also be crucial in our attack.

Since inversions are typically much more expensive than multiplications, several types of projective coordinate systems have been developed. Standard projective coordinates [5] represent an elliptic curve point $P = (x, y)$ by (X, Y, Z) where $x = X/Z$ and $y = Y/Z$, whereas Jacobian projective coordinates [5] use $x = X/Z^2$ and $y = Y/Z^3$. The above addition/doubling formulae can easily be reformulated using projective coordinates, but the resulting formulae will also depend on a only.

Group law implementation An implementer of an elliptic curve system is not only faced with the choice of the elliptic curve model to use, such as short Weierstrass, Montgomery [34], Edwards [7, 8], Hessian [39], etc., and the choice of an appropriate coordinate system like projective [5] or Jacobian [5], but also with the handling of borderline cases. Indeed, the above addition formula (1) can only handle the cases where $P \neq \pm Q$, $P \neq \mathcal{O}$ and $Q \neq \mathcal{O}$. Similarly, the doubling formula (2) will fail when P is a point of order two or $P = \mathcal{O}$.

The way in which the implementation handles these borderline cases leads to the following classification: *full and partial domain correctness.* In the full domain correctness case, the implementation computes $P + Q$ and $2 \cdot P$ correctly for all P, Q. In the partial domain correctness case the implementation either stops working (e.g. division by zero occurs), computes on invalid points or ends in a fixed point (both cases occur when using the above formulae in projective or Jacobian coordinates, see Table 3 in the appendix).

2.2 Scalar multiplication

The basic operation in classical cryptosystems such as RSA and ECC is exponentiation in the underlying group. For elliptic curves, this exponentiation is called scalar multiplication since given a point P and a scalar k, it computes $k \cdot P$ by repeatedly using the double/add operations.

The most basic scalar multiplication algorithm is the binary double-and-add algorithm, which computes $k \cdot P$ according to the binary expansion of $k = \sum_{i=0}^{n-1} k_i 2^i$. Depending on the direction in which the bits of k are scanned, we obtain a left-to-right or right-to-left variant.

The left-to-right variant is described in Algorithm 1 and will be used to illustrate our attack. The applicability of our attack to SSCA and DSCA resistant scalar multiplication algorithms will be discussed in Section 5.

Algorithm 1: Double and Add Left-to-Right

```
Input: $\boldsymbol{P}$, $k = (k_{n-1}, k_{n-2}, \ldots, k_0)_2$
Output: $\boldsymbol{Q} = k \cdot \boldsymbol{P}$
$\boldsymbol{R} \leftarrow P$ ;
for $i \leftarrow n-2$ down to 0 do
    $\boldsymbol{R} \leftarrow 2 \cdot \boldsymbol{R}$ ;
    if $(k_i = 1)$ then $\boldsymbol{R} \leftarrow \boldsymbol{R} + \boldsymbol{P}$ ;
end
return $\boldsymbol{R}$
```

3 Elliptic curve points with low order neighbours

In this section, we consider the following problem, the solution of which is crucial for our attack: given an elliptic curve $E : y^2 = x^3 + ax + b$ over $\mathbb{F}_p$, two integers

ℓ and Δ, is it possible to construct a point $P := (x_P, y_P)$ in $E(\mathbb{F}_p)$ with the following properties:

- there exists a curve $E' : y^2 = x^3 + ax + b'$ over $\mathbb{F}_p$
- with a point $P' = (x_{P'}, y_{P'}) \in E'(\mathbb{F}_p)$ of order ℓ
- such that the Hamming distance of the bit-representations $x_P||y_P$ and $x_{P'}||y_{P'}$ equals Δ.

When $\Delta = 1$, i.e. the coordinates differ in a single bit, we call the points P and P' *neighbours*. We will describe an effective construction of points P with neighbours P' of a given order ℓ and with $x_P = x_{P'}$, i.e. the bit-flip occurred in the y-coordinate only. The construction can be easily extended to encompass bit-flips in x_P, and indirect neighbours, i.e. $\Delta > 1$.

In Section 3.1 we first show how to construct points of given order and in Section 3.2 we adapt this method to find points with low order neighbours.

3.1 Constructing points of given order

Given an elliptic curve E over $\mathbb{F}_p$, we can consider the points on E of order dividing n, i.e. points $P \in E(\overline{\mathbb{F}}_p)$ with $n \cdot P = \mathcal{O}$, where the coordinates of P can lie in any extension field of $\mathbb{F}_p$. These points can be characterized explicitly using the so called division polynomials [5]. For $n \in \mathbb{N}$ define polynomials $\psi_n(x, y)$ recursively as follows:

$$\begin{aligned}
\psi_0 &= 0,\ \psi_1 = 1, \psi_2 = 2y,\ \psi_3 = 3x^4 + 6ax^2 + 12bx - a^2, \\
\psi_4 &= 4y(x^6 + 5ax^4 + 20bx^3 - 5a^2x^2 - 4abx - 8b^2 - a^3), \\
\psi_{2m+1} &= \psi_{m+2}\psi_m^3 - \psi_{m-1}\psi_{m+1}^3, m \geq 2, \\
\psi_{2m} &= \frac{\psi_m(\psi_{m+2}\psi_{m-1}^2 - \psi_{m-2}\psi_{m+1}^2)}{2y}, m \geq 3\,.
\end{aligned}$$

For any point $P \in E(\overline{\mathbb{F}}_p)$ with $P \neq \mathcal{O}$, we then have that $n \cdot P = \mathcal{O}$ if and only if $\psi_n(x_P, y_P) = 0$. Furthermore, one can show by induction on n that ψ_n for n odd and $\psi_n/2y$ for n even, are polynomials in x only. We denote these polynomials by $\phi_n(x)$. It is easy to see that $n \cdot P = \mathcal{O}$ and $2 \cdot P \neq \mathcal{O}$ if and only if $\phi_n(x_P) = 0$.

3.2 Constructing points with low order neighbours

Given an elliptic curve E over $\mathbb{F}_p$ and an integer ℓ, we want to construct a point $P = (x_P, y_P)$ in $E(\mathbb{F}_p)$ with neighbour $P' = (x_P, y_P \oplus \epsilon)$ with $\epsilon = 2^k$ for some $k < \log_2(p)$ and $\ell \cdot P' = \mathcal{O}$. For $\ell > 2$, the points P and P' therefore have to satisfy the following non-linear system of equations:

$$\begin{cases}
\boldsymbol{y}_P{}^2 - \boldsymbol{x}_P{}^3 - a \cdot \boldsymbol{x}_P - b = 0 & P \in E(\mathbb{F}_p) \\
(\boldsymbol{y}_P \oplus \epsilon)^2 - \boldsymbol{x}_P{}^3 - a \cdot \boldsymbol{x}_P - \boldsymbol{b'} = 0 & P' \in E'(\mathbb{F}_p) \\
\phi_\ell^{a,\boldsymbol{b'}}(\boldsymbol{x}_P) = 0 & \ell \cdot P' = \mathcal{O},
\end{cases}$$

where the unknown variables are printed in bold face. Since the $\oplus$-operation is not very algebraic, we will consider the following two cases that lead to equivalent results, namely we replace $y_p \oplus \epsilon$ by $y_p \pm \epsilon$ and then verify afterwards if an actual bit-flip occurred, i.e. that there was no carry.

Subtracting the first two equations expresses b' as a function of y_P, namely, $b' = \pm 2\epsilon y_P + \epsilon^2 + b$. Substituting this expression in the last equation leads to a bivariate polynomial in x_P and y_P, which we call $\Upsilon_\ell(x_P, y_P)$. The points P for the given ℓ and ϵ therefore are solutions of $E(x_P, y_P) = 0$ and $\Upsilon_\ell(x_P, y_P) = 0$. These solutions can be easily found by a Groebner basis [13] computation or by taking the resultant

$$R(x_p) = \mathrm{Resultant}_{y_P}(E(x_P, y_P), \Upsilon_\ell(x_P, y_P)),$$

finding all possibilities for x_P as roots of R over $\mathbb{F}_p$ and the corresponding y_P from $E(x_P, y_P) = 0$. A final check is then necessary to only retain those (x_P, y_P) where the $\pm$ operation actually caused a bit-flip, in particular, in the $+\epsilon$-case (resp. $-\epsilon$-case) we only retain those results where the k-th bit is zero (resp. one).

To analyze the complexity of solving the above system, we simply need to figure out the degree of $\phi_\ell^{a,b'}$ in x_P and b'. The degree in x_P is easily seen to be $(\ell^2 - 1)/2$ since the full ℓ-torsion contains ℓ^2 points. The degree in b' can be seen to be upperbounded by $(\ell^2 - 1)/6$ since the same recursion holds and the degree in b' is three times smaller than for x_P in the initializations. This leads to a resultant of degree $\ell^2 - 1$ and an overall complexity of $\tilde{O}(\ell^4)$ to solve the non-linear system of equations.

The probability that a given curve admits a point with neighbour of order ℓ and bit-flip in position k (note that both ℓ and k are fixed) can be roughly approximated as follows. Denote by $P(n, p)$ the probability that a random polynomial of degree n has at least one root in $\mathbb{F}_p$. Assuming we can consider the resultant R as a random polynomial, then the probability is roughly the product of:

– the probability $P(\ell^2 - 1, p)$ that R has at least one root x_P in $\mathbb{F}_p$,
– the probability $1/2$ that the corresponding y_P is in $\mathbb{F}_p$ (and when it does, there are two roots y_P),
– the probability $3/4$ that of these two roots, at least one has a bit-flip in position k.

Note that we only analyze the case of $+\epsilon$ and not also $-\epsilon$, since the solvability of both systems of equations is not really independent. The overall probability therefore is roughly $3/8 \cdot P(\ell^2 - 1, p)$. A closed expression for $P(n, p)$ exists [32] and this can easily be shown to satisfy $P(n, p) > 1/2$, which leads to a lower bound of $3/16$. Note that this high probability stands in stark contrast with the probability that a given fixed point has a neighbour of order ℓ, which we expect to be in $O(1/p)$.

The above algorithm can be easily extended to any given fixed error pattern, such as multiple bit-flips, or setting certain bits to zero/one. Furthermore, errors in the x-coordinate can also be dealt with.

To illustrate the effectiveness of the above procedure, in Table 1 we provide several example points with low order neighbours for the NIST P-192 curve [35], i.e. the curve over $\mathbb{F}_p$ with $p = 2^{192} - 2^{64} - 1$, $a = -3$ and $b =$ 0x64210519E59C80E70FA7E9AB72243049FEB8DEECC146B9B1. For each small integer ℓ, the table gives a P with neighbour of order ℓ when a specific bit of the y-coordinate is flipped (bit 0 is the LSB). Each of these examples was generated in less than a second using Magma [11] on a standard laptop.

Table 1. Points with neighbours of low order on NIST P-192 curve.

Order	P	bit-flip

2	xP = 0x6D9D789820A2C19237C96AD4B8D86B87FB49D4D6C728B84F
	yP = 0x1
		0

3	xP = 0x8E1AEBDD6009F114490C7BC2C02509F8E432ED15F10C2D33
	yP = 0x7A568946EFA602B3624A61E513E57869CAF2AE854E1A17B
		2

4	xP = 0xB317D7BBD023E6293F1506221F5BC4A23D4BE2E05328C5F7
	yP = 0xC70D48794F409831097620C0865B7D567329728C634CA6AE
		0

5	xP = 0xCC9BCC0061F64371E3C3BDE165DAD5380A7DC1919765940
	yP = 0xCC8B36B37928334B8AFD7A9FCCFB4B0773E94A4178093458
		8

6	xP = 0xC3F76445E6A52138E283E485092F005BE0821C3F9E96B05E
	yP = 0x535DBCCB593D72E7885B66E57FD13A8FF9C57A8F8B91CE48
		1

7	xP = 0x5C003567728CCBC9F4C06620B9973193837BAEC67A29E43A
	yP = 0x408D0C3135006B03EFF80961394D890F0E86D9FD1BA4EEC6
		3

8	xP = 0x74FD6A1AD39479C75A85305FA786E1DBDC845E03754E723E
	yP = 0x6EF58ABFC0B71047BA4F425652B3EC1746EBE8FE16FEA1F5
		1

4 Combined attack using low order neighbours

In this section we introduce a new combined attack using points with low order neighbours. The system under attack is the following: we have access to a target implementation that on input an elliptic curve point P computes $k \cdot P$ for some unknown secret k. The goal is to recover the secret k.

The basic version of our attack requires the following *two assumptions*. The realistic nature of these assumptions and the applicability of our attack will be analyzed in Section 5.

1. It is possible, e.g. using side channel information, to determine when an intermediate result in the computation becomes $\mathcal{O}$.
2. It is possible to inject a fault immediately after initial validity checks, resulting in a bit-flip in a predetermined position.

The attack then proceeds as follows: we input a point P with low order neighbour P' and, after the initial validity checks have passed, inject a fault that turns P into P'. The implementation then tries to compute $k \cdot P'$. Since P' has low order, it is highly likely that an intermediate computation will result in $\mathcal{O}$. This corresponds to the fact that the part of the secret scalar k that has been processed up to that point is divisible by the order of P'.

If and how the implementation continues to run depends solely on how the elliptic curve group operations are implemented, i.e. whether the implementation is full or partial domain correct.

4.1 Full domain correctness

The implementation will compute the scalar multiplication $k \cdot P'$ until the final validity checks, at which point it will abort since $k \cdot P'$ is not on the curve E. During the computation however, we will obtain a huge amount of information of the following form: assume the order of P' is ℓ, then every time an elliptic curve addition/doubling results in $\mathcal{O}$, we know that the part of the scalar processed up to that point is divisible by ℓ. Note that we also obtain extra information when $\mathcal{O}$ does not appear, since then the corresponding part of the scalar is not divisible by ℓ.

This attack is extremely powerful since in most cases one trace will suffice to recover (almost all of) k. In Section 5.3 we will show that the attack can recover ephemeral keys, blinded keys and randomly split keys.

Example To illustrate the effectiveness of this attack in the full domain correctness case, we apply it to an implementation using Algorithm 1. If we choose to input a point P with neighbour P' of order 2, all occurring computations $(2 \cdot P', 2 \cdot \mathcal{O}, \mathcal{O} + P')$ are borderline cases, which may not be desirable. Therefore, we choose a point P with neighbour P' of order 4.

The computation of $\boldsymbol{R} \leftarrow 2 \cdot \boldsymbol{R}$ then either consists of $2 \cdot P'$, $2 \cdot (2P')$, $2 \cdot (3P')$ or $2 \cdot \mathcal{O}$. Note that the cases $2 \cdot (2P')$ and $2 \cdot \mathcal{O}$ are borderline cases, and thus distinguishable from the cases $2 \cdot P'$ and $2 \cdot (3P')$, which are ordinary doublings. The crucial point to note is that point addition always generates odd multiples of P' and thus will never result in $\mathcal{O}$. Furthermore, since P' has order 4, the point $\mathcal{O}$ will only occur after two consecutive doublings. Therefore, if $\mathcal{O}$ occurs during the processing of bit k_i, we know that bit k_{i+1} must have been zero. This uniquely identifies the zero key bits (except for possibly the LSB), which implies that the other key bits have to be one. As such, we easily obtain all of k with one trace only.

Table 2 shows the intermediate results for the computation of $k \cdot P'$ where $k = 5405$ and $\ell = 4$. Note that we assume that distinguishing point addition from point doubling is not possible. As such, the adversary sees a sequence of normal operations (additions or ordinary doublings), denoted by `Op`, and occurrences of $\mathcal{O}$ as shown in the fourth row of Table 2. To recover the secret key, the adversary proceeds as follows: in step one, he puts a 0 in each cell to the left of an $\mathcal{O}$. Then in step two, he groups the empty cells in pairs of two, from left to right, merges them and writes a 1 in the resulting cell. If there is a single cell left in the end, he writes a zero in it.

Table 2. Intermediate results in the computation of $5405 \cdot P'$ with $\ell = 4$ and view of the adversary when attacking the scalar multiplication.

i	11	10	9	8	7	6	5	4	3	2	1	0
k_i	0	1	0	1	0	0	0	1	1	1	0	1
$\boldsymbol{R}$	$2P'$	$\mathcal{O}$,P'	$2P'$	$\mathcal{O}$,P'	$2P'$	$\mathcal{O}$	$\mathcal{O}$	$\mathcal{O}$,P'	$2P'$,$3P'$	$2P'$,$3P'$	$2P'$	$\mathcal{O}$,P'

view	`Op`	$\mathcal{O}$	`Op`	`Op`	$\mathcal{O}$	`Op`	`Op`	$\mathcal{O}$	$\mathcal{O}$	$\mathcal{O}$	`Op`	`Op`	`Op`	`Op`	`Op`	`Op`	$\mathcal{O}$	`Op`
step 1	0			0			0	0	0							0		
step 2	0	1		0	1		0	0	0	1		1		1		0	1	

4.2 Partial domain correctness

Partial domain correctness implies that we can only gather information up to the first occurrence of the point $\mathcal{O}$. Indeed, either the implementation simply crashes during the computation of $\mathcal{O}$ or it performs some nonsensical computations thereafter. The result is that for each point submitted, we can only obtain partial information about k. When k is fixed over several invocations, this is not a real problem since we can submit many points with neighbours of different order and then deduce all bits of k from this information. Note that due to the behavior of the implementation, i.e. no further information after occurrence of $\mathcal{O}$, the orders of the neighbouring points submitted do not have to be coprime.

The type of information gathered will be of the following form: let $k = \sum_{i=0}^{n-1} k_i 2^i$, then for each small integer ℓ we will obtain the index $I(\ell)$ such that the leftmost (or rightmost) $I(\ell)$ bits of k form an integer divisible by ℓ. By definition we set $I(\ell) = 0$ when no part of k is divisible by ℓ. As such we obtain a list of positive information `PosInfo`, consisting of pairs $[\ell, I(\ell)]$, and a list of negative information `NegInfo` containing those ℓ for which each $I(\ell) = 0$. The list `PosInfo` will be sorted according to $I(\ell)$.

A very simple incremental search algorithm is given in Algorithm 2. The algorithm keeps a list `PartialKeys` containing all possibilities for the `BitsScanned` leftmost bits of k. The procedure `ExpandPartialKeys` expands all partial keys in the list by appending (on the right) all possible bit sequences of length `PosInfo[j][2] - BitsScanned` and then only keeps those candidates divisible by `PosInfo[j][1]`. It furthermore updates `BitsScanned` to `PosInfo[j][2]`. The function `PrunePartialKeys` simply removes all elements from `PartialKeys` that violate one of the non-divisibility conditions for any of the integers in `NegInfo`. We implemented this algorithm in Magma and ran several tests to evaluate its behaviour for the NIST P-192 curve. Given a fixed secret random k, we computed for each integer ℓ smaller than an upper bound B the value $I(\ell)$ and then tried to recover k from `PosInfo` and `NegInfo`. The tests show that even for $B \simeq 100$ we can typically recover a large part of the secret k (much more than 100 bits on average) and that for larger values of B like 192 or 384 we recover most of k bar a few least significant bits.

Algorithm 2: Recovering private key from `PosInfo` and `NegInfo`.

```
PartialKeys ← ∅, BitsScanned ← 0
for $j$ from 1 upto # PosInfo do
    ExpandPartialKeys (& PartialKeys, PosInfo[j], & BitsScanned)
    PrunePartialKeys (& PartialKeys, NegInfo)
end
return PartialKeys, BitsScanned.
```

5 Analysis of the attack

In this section we discuss the assumptions made in the previous section and analyze the attack for a wide range of implementation choices, such as coordinate systems and curves used, scalar multiplication algorithms and finally, common countermeasures against SSCA and DSCA attacks and validity checks against fault attacks.

5.1 Analysis of assumptions

Chosen input point The target implementation is assumed to compute $k \cdot P$ for any given input point P, where k is supposed to be secret. This setting arises for instance in ElGamal decryption [22], ECIES [1] and in static Diffie-Hellman key agreement [21]. In the latter case, one of the ephemeral keys is simply the long term public key. We note that the attack does not apply to ECDSA [41], where the ephemeral key is computed on a fixed base point P (unless P has neighbours of low order).

Recognizing $\mathcal{O}$ via side channels In the case of partial domain correctness, the implementation either crashes during the computation of $\mathcal{O}$ or it ends up in $\mathcal{O}$ and remains there. We assume that either event can be detected through side channels. Indeed, if the implementation crashes it can for example stall or exit the scalar multiplication routine early, which should be clearly visible e.g. in power traces. If the implementation continues to run it will get stuck in $\mathcal{O}$, which should be visible as a repetitive pattern.

In the case of full domain correctness, the implementation does not crash because it correctly deals with all borderline cases. Most textbooks on ECC, e.g. Hankerson et al. [26], use checks and conditional branches in their code examples to ensure full domain correctness. It is well known that conditional branches can leak through side channels [31, 17] and so it is clear that the occurrence of any borderline case can be easily detected. Even if we assume that these checks and branches are implemented with side channel resistance in mind (which is highly unlikely) the actual occurrence of $\mathcal{O}$ (a point with at least one coordinate equal to zero) in a point operation should be visible [2].

Fault injection The assumption that an adversary can flip a single chosen bit in an implementation is certainly strong. We can relax this assumption greatly using a trivial approach: by repeatedly faulting a specified byte (resulting in a random byte), after an average of 256 trials, the fault will be precisely the fault required. With overwhelming probability only the required fault will lead to a point of low order, thus the good case is easily distinguished from undesired faults. However, we still have to assume that an adversary can inject a fault with sufficiently precise timing, in this case after initial validity checks.

The construction of points with low order neighbours is also flexible enough to accommodate a more accurate fault model for the target implementation. Assume we have extra information on the most likely state of a byte after fault injection, then we can compute points specially crafted for this fault pattern.

Group law formulae An implicit assumption, which is automatically satisfied when using the formulae given in Eqs. (1) and (2), is that the group law formulae do not depend on *all* coefficients of the curve equation. More formally, assume the elliptic curve equation $E(a_1, \ldots, a_k)$ depends on k coefficients, but that only the first $m < k$ appear in the group law formulae. Then the implementation can also be used to compute correctly on all elliptic curves with the same $a_1, \ldots, a_m$, but differing $a_{m+1}, \ldots, a_k$. Note that for *all* elliptic curve forms, it is always possible to write down group law formulae with $m < k$. However, for the most efficient formulae used in practice, our assumption seems only valid for Weierstrass forms, which are most widely used, and Hessian forms. In fact, using group law formulae involving all coefficients of the curve combined with initial and final validity checks, is a possible combination of countermeasures to our attack.

5.2 Scalar multiplication

Many scalar multiplication algorithms have been proposed, either to speed up the computation or to aid resistance against simple side channel analysis. In this section we focus on scalar recoding [5], the Montgomery powering ladder [28], unified formulae [12, 19] and side channel atomicity [14].

Scalar representation Apart from the usual binary representation of the scalar $k = \sum_{i=0}^{n-1} k_i 2^i$, several other representations are frequently used. The non-adjacent form (NAF) represents $k = \sum_{i=0}^{n-1} k_i 2^i$, where $k_i \in \{0, \pm 1\}$. More generally, a width w-NAF of an integer k is an expression $k = \sum_{i=0}^{n-1} k_i 2^i$ with each nonzero k_i odd, $|k_i| < 2^{w-1}$, $k_{n-1} \neq 0$ and at most one of any w consecutive digits nonzero. In all cases, we still obtain a similar type of information as in the basic attack: when $\mathcal{O}$ is encountered, we know that the part of the scalar processed up to that point is divisible by ℓ. However, since the number of intermediate points computed during the scalar multiplication is no longer n but n/w, the probability of hitting $\mathcal{O}$ is lower.

Montgomery powering ladder The Montgomery powering ladder given in Algorithm 3 is a popular choice because it provides speed and a highly regular structure.

Algorithm 3: Montgomery powering ladder

```
Input: $\boldsymbol{P}$, $k = (k_{n-1}, k_{n-2}, \ldots, k_0)_2$
Output: $\boldsymbol{Q} = k \cdot \boldsymbol{P}$
$\boldsymbol{R}_0 \leftarrow P$, $\boldsymbol{R}_1 \leftarrow 2 \cdot P$ ;
for $i \leftarrow n-2$ down to 0 do
    $\boldsymbol{R}_{\neg k_i} \leftarrow \boldsymbol{R}_{k_i} + \boldsymbol{R}_{\neg k_i}$, $\boldsymbol{R}_{k_i} \leftarrow 2 \cdot \boldsymbol{R}_{k_i}$ ;
end
return $\boldsymbol{R}_0$
```

Attacking the Montgomery ladder is a bit more tricky because the sequence of operations is fixed and independent of the key. Nevertheless, the attack applies since it does not exploit the sequence of operations but the evolution of the intermediate values. Assume we input a point P with neighbour P' of order 4 and inject a fault after initial validity checks. The implementation will then try to compute $k \cdot P'$. Note that if two consecutive bits of k are equal, then the same point (either $\boldsymbol{R}_0$ or $\boldsymbol{R}_1$) will be doubled twice by the operation $\boldsymbol{R}_{k_i} \leftarrow 2 \cdot \boldsymbol{R}_{k_i}$ resulting in $\mathcal{O}$. On the other hand, if two consecutive bits differ, an ordinary doubling $2 \cdot P'$ or $2 \cdot (3P')$ will be computed. Finally, note that $\mathcal{O}$ can never be the result of the addition operation $\boldsymbol{R}_{k_i} + \boldsymbol{R}_{\neg k_i}$, since this is always an odd multiple of P'. As such, we obtain (almost all of) k with one trace only in the full domain correctness case.

Unified formulae and side channel atomicity These countermeasures render point additions and doublings indistinguishable to prevent SSCA, and they can be implemented together with a possibly faster, irregular scalar multiplication algorithm like double-and-add (at the cost of leaking the Hamming weight of the exponent). It is clear that our attack is not affected by countermeasures of this kind because it does not require point operations to be distinguishable.

5.3 Common DSCA and FA countermeasures

Random scalar splitting [15] With this countermeasure, the scalar k is randomly split into two parts: $k = k_1 + k_2$. As such, $Q = k \cdot P$ can be computed as $k_1 \cdot P + k_2 \cdot P$ by two consecutive scalar multiplications and addition of their results. In the case of full domain correctness, (almost all of) k_1 and k_2 can be revealed (assuming that only the final output point is checked for validity), which immediately results in (a small number of candidates for) k. Otherwise the situation is similar to that of partial domain correctness. In the partial domain correctness case, the attack will no longer work, since we will only be able to recover a part of k_1 or k_2, but not both at the same time. Indeed, typically the implementation will stop working the first time it hits $\mathcal{O}$.

Scalar randomization [20] In this case, the scalar is blinded using a multiple of the curve order, i.e. k is replaced by $k' = k + r \cdot \#E$. It is easy to see that this countermeasure is useless in the full domain correctness case, where only a single trace is needed. For partial domain correctness, we do get partial information on k', but currently have no method to exploit this. Note that the same conclusion applies for ephemeral keys.

Coordinate randomization [20] This countermeasure assumes that some form of projective coordinates are being used and that the coordinates of the input point P are randomized before the scalar multiplication is started. For instance, when using projective coordinates, $P = (rX_P, rY_P, rZ_P)$ with r randomly chosen is used. It is easy to see that our attack remains valid if initial checks are performed *before* point randomization.

Random elliptic curve isomorphisms [27] This method first applies a random isomorphism of the form $\psi : (x, y) \mapsto (r^2x, r^3y)$ and then proceeds by computing $Q = k \cdot \psi(P)$ and outputting $\psi^{-1}(Q)$. Since an isomorphism does not change the order of a point, it is clear that the attack still applies if initial checks are performed *before* ψ is applied.

Isogeny defence [40] To prevent Goubin's attack using special points of large order, Smart proposed to use an isogeny $\mathcal{I}$ to map the input points to an isogenous curve without special points. Furthermore, for each curve in the main standards Smart provides a fixed isogeny that works for that curve. It is clear that our attack still applies if we look for points P with low order neighbours on the isogenous curve instead of on the original curve. The input to the target device will then be the points $\mathcal{I}^{-1}(P)$ and the fault will be injected after initial checks and isogeny have been applied.

Point blinding [20] With this countermeasure, the implementation contains a random point R and the corresponding multiple $k \cdot R$. The scalar multiplication $k \cdot P$ is computed by first computing $k \cdot (P + R)$ and then subtracting $k \cdot R$ from the result. Since we have no control over the point R, we cannot compute an appropriate point P such that we can fault the point $P + R$ into a point of low order. As such, this countermeasure does thwart our attack, both in the full and partial domain correctness case. Point blinding can be seen as an instance of infective computation [42].

Cofactor multiplication [40] To prevent small subgroup attacks, most protocols can be reformulated using cofactor multiplication. For instance, the Diffie-Hellman protocol can be adapted as follows: a user first computes $Q \leftarrow h \cdot P$

and then $R \leftarrow k \cdot Q$ if $Q \neq \mathcal{O}$. It is easy to see that our attack still applies when we input a point with neighbour of order different from h.

Validity checks [9, 15] To prevent fault attacks, Biehl et al. [9] and Ciet and Joye [15] recommend input/output point validity checks and curve parameter checks. These recommendations were part of the original motivation for our work and do not prevent the attack.

5.4 Curves over finite fields of characteristic two

Although the attack has mainly been described for elliptic curves in Weierstrass form over fields of large characteristic, we briefly touch on the characteristic two case. The short Weierstrass form is given by $E : y^2 + xy = x^3 + ax^2 + b$. The applicability of our attack then depends on the coordinate system being used. For affine and standard projective coordinates, the attack applies since only the a-coefficient is used in the group law formulae. For Jacobian coordinates the attack does not apply since both a and b are used in the group law formulae. For Lopez-Dahab formulae, only b is used in the group law formulae, but changing a only results in an isomorphic curve or its quadratic twist. As such it is impossible to find a point of given low order, since both the curve and its twist should not have many small subgroups.

6 Conclusions

We have described a novel attack that combines three ideas: fault injection, simple side channel analysis, and specially crafted, but valid input points that after a single fault injection have very low order. Our attack breaks ECC implementations that are protected by many of the known countermeasures such as initial and final point validity checks, curve parameter checks, cofactor multiplication check, SSCA countermeasures and bypasses many DSCA countermeasures. A secondary yet irritating result of our analysis is that proper, i.e. full domain correct implementations are more vulnerable to the attack and can be broken using one successful fault injection.

The attack does not apply to protocols that use a fixed point P (with no near neighbours of low order). For other applications, the attack can be prevented by physical fault injection sensors, concurrent point validity checks, using group law formulae that involve all curve coefficients, using randomized coordinates or randomized curve isomorphisms with randomization *before* the initial point validity check and by point blinding.

References

1. M. Abdalla, M. Bellare, and P. Rogaway. DHAES: An encryption scheme based on the Diffie-Hellman problem. Submission to P1363a : Standard specifications for Public-Key-Cryptography: Additional techniques, 2000.

2. T. Akishita and T. Takagi. Zero-value point attacks on elliptic curve cryptosystem. In *ISC 2003*, volume 2851 of *LNCS*, pp. 218–233. Springer-Verlag, 2003.
3. F. Amiel, K. Villegas, B. Feix, and L. Marcel. Passive and active combined attacks: Combining fault attacks and side channel analysis. In *FDTC 2007*, pp. 92–102. IEEE Computer Society, 2007.
4. R. Anderson and M. Kuhn. Tamper resistance - a cautionary note. In *The Second USENIX Workshop on Electronic Commerce Proceedings*, pages 1–11. USENIX Association, 1996.
5. R. M. Avanzi, H. Cohen, C. Doche, G. Frey, T. Lange, K. Nguyen, and F. Vercauteren. *Handbook of elliptic and hyperelliptic curve cryptography.* Discrete Mathematics and Its Applications. Chapman & Hall/CRC, 2006.
6. H. Bar-El, H. Choukri, D. Naccache, M. Tunstall, and C. Whelan. The sorcerer's apprentice guide to fault attacks. *Proceedings of the IEEE*, 94(2):370–382, 2006.
7. D. J. Bernstein and T. Lange. Faster addition and doubling on elliptic curves. In Kaoru Kurosawa, editor, *ASIACRYPT 2007*, volume 4833 of *LNCS*, pp. 29–50. Springer-Verlag, 2007.
8. D. J. Bernstein, T. Lange, and R. R. Farashahi. Binary edwards curves. In *CHES 2008*, volume 5154 of *LNCS*, pp. 244–265. Springer-Verlag, 2008.
9. I. Biehl, B. Meyer, and V. Müller. Differential fault attacks on elliptic curve cryptosystems. In *CRYPTO 2000*, volume 1880 of *LNCS*, pp. 131–146. Springer-Verlag, 2000.
10. D. Boneh, R. A. DeMillo, and R. J. Lipton. On the importance of checking cryptographic protocols for faults (extended abstract). In *EUROCRYPT 1997*, volume 1233 of *LNCS*, pp. 37–51. Springer-Verlag, 1997.
11. W. Bosma, J. Cannon and C. Playoust. The Magma algebra system. I. The user language. In *J. Symbolic Comput.*, 24(3-4), pp. 235–265, 1997.
12. E. Brier and M. Joye. Weierstraß elliptic curves and side-channel attacks. In *PKC 2002*, volume 2274 of *LNCS*, pp. 335–345. Springer-Verlag, 2002.
13. B. Buchberger. *Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal.* PhD thesis, Universität Innsbruck, 1965.
14. B. Chevallier-Mames, M. Ciet, and M. Joye. Low-cost solutions for preventing simple side-channel analysis: Side-channel atomicity. *IEEE Trans. Computers*, 6(53):760–768, 2004.
15. M. Ciet and M. Joye. (Virtually) free randomization techniques for elliptic curve cryptography. In *ICICS 2003*, volume 2836 of *LNCS*, pp. 348–359. Springer-Verlag, 2003.
16. M. Ciet and M. Joye. Elliptic curve cryptosystems in the presence of permanent and transient faults. *Designs, Codes and Cryptography*, 36(1):33–43, 2005.
17. C. Clavier and J-S. Coron. On the implementation of a fast prime generation algorithm. In *CHES 2007*, volume 4727 of *LNCS*, pp. 443–449. Springer-Verlag, 2007.
18. C. Clavier, B. Feix, G. Gagnerot, M. Roussellet, and V. Verneuil. Horizontal correlation analysis on exponentiation. In *ICICS 2010*, volume 6476 of *LNCS*, pp. 46–61. Springer-Verlag, 2010.
19. C. Clavier and M. Joye. Universal exponentiation algorithm. In *CHES 2001*, volume 2162 of *LNCS*, pp. 300–308. Springer-Verlag, 2001.
20. J-S. Coron. Resistance against differential power analysis for elliptic curve cryptosystems. In *CHES 1999*, volume 1717 of *LNCS*, pp. 292–302. Springer-Verlag, 1999.

21. W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Trans. Inform. Theory*, 22(6):644–654, 1976.
22. T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *CRYPTO 1984*, volume 196 of *LNCS*, pp. 10–18. Springer-Verlag, 1985.
23. J. Fan, X. Guo, E. De Mulder, P. Schaumont, B. Preneel, and I. Verbauwhede. State-of-the-art of secure ECC implementations: A survey on known side-channel attacks and countermeasures. In *HOST 2010*, pages 76–87, 2010.
24. K. Gandolfi, C. Mourtel, and F. Olivier. Electromagnetic analysis: Concrete results. In *CHES 2001*, volume 2162 of *LNCS*, pp. 251–261. Springer-Verlag, 2001.
25. L. Goubin. A refined power-analysis attack on elliptic curve cryptosystems. In *PKC 2003*, volume 2567 of *LNCS*, pp. 199–210. Springer-Verlag, 2003.
26. D. Hankerson, A. J. Menezes, and S. Vanstone. *Guide to Elliptic Curve Cryptography.* Springer-Verlag, 2004.
27. M. Joye and C. Tymen. Protections against differential analysis for elliptic curve cryptography. In *CHES 2001*, volume 2162 of *LNCS*, pp. 377–390. Springer-Verlag, 2001.
28. M. Joye and S.-M. Yen. The Montgomery powering ladder. In *CHES 2002*, volume 2523 of *LNCS*, pp. 291–302, Springer-Verlag, 2002.
29. N. Koblitz. Elliptic curve cryptosystem. *Math. Comp.*, 48:203–209, 1987.
30. P. C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In *CRYPTO 1996*, volume 1109 of *LNCS*, ppİ04–113. Springer-Verlag, 1996.
31. P. C. Kocher, J. Jaffe, and B. Jun. Differential power analysis. In *CRYPTO 1999*, volume 1666 of *LNCS*, pp. 388–397. Springer-Verlag, 1999.
32. V. K. Leontév. Roots of Random Polynomials over a Finite Field. In *Mat. Zametki*, 80:2 (2006), pp. 313–316.
33. Victor S. Miller. Uses of elliptic curves in cryptography. In *CRYPTO 1985*, volume 218 of *LNCS*, pp. 417–426. Springer-Verlag, 1986.
34. P. L. Montgomery. Speeding up the Pollard and elliptic curve methods for factorizations. *Mathematics of Computation*, 48:243–264, 1987.
35. National Institute of Standards and Technology (NIST). Digital signature standard (DSS), FIPS PUB 186-3, 2009.
36. J.-J. Quisquater and D. Samyde. Electromagnetic analysis (EMA): Measures and counter-measures for smart cards. In *E-smart 2001*, volume 2140 of *LNCS*, pp. 200–210. Springer-Verlag, 2001.
37. R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21:120–126, 1978.
38. S. P. Skorobogatov and R. J. Anderson. Optical fault induction attacks. In *CHES 2002*, volume 2523 of *LNCS*, pp. 2–12, Springer-Verlag, 2002.
39. N. P. Smart. The Hessian form of an elliptic curve. In *CHES 2001*, volume 2162 of *LNCS*, pp. 118–125. Springer-Verlag, 2001.
40. N. P. Smart. An analysis of goubin's refined power analysis attack. In *CHES 2003*, volume 2779 of *LNCS*, pp. 281–290. Springer-Verlag, 2003.
41. S. Vanstone. Responses to NIST's proposal. *Communications of the ACM*, 35:50–52, 1992.
42. S-M. Yen, S. Kim, S. Lim, and S-J. Moon. RSA speedup with residue number system immune against hardware fault cryptanalysis. In *ICISC 2001*, volume 2288 of *LNCS*, pp. 397–413. Springer-Verlag, 2001.

Appendix

Table 3. Borderline cases for projective and Jacobian coordinates.

$E(\mathbb{F}_p)$: $y^2 = x^3 + ax + b$					
Coordinate System	Operation	Using a	Using b	Input	Output
Projective	PA($\boldsymbol{P_1}, \boldsymbol{P_2}$)	-	-	$\boldsymbol{P_1}$=$\boldsymbol{P_2}$	(0,0,0)
				$\boldsymbol{P_1}$=-$\boldsymbol{P_2}$	(0,*,0)
				$\boldsymbol{P_1}$=(0,*,0)	(0,0,0)
	PD($\boldsymbol{P_1}$)	+	-	Order($\boldsymbol{P_1}$)=2	(0,*,0)
				$\boldsymbol{P_1}$=(0,*,0)	(0,0,0)
Jacobian	PA($\boldsymbol{P_1}, \boldsymbol{P_2}$)	-	-	$\boldsymbol{P_1}$=$\boldsymbol{P_2}$	(0,0,0)
				$\boldsymbol{P_1}$=-$\boldsymbol{P_2}$	(*,*,0)
				$\boldsymbol{P_1}$=(*,*,0)	(*,*,0)
				$\boldsymbol{P_1}$=(0,0,0)	(0,0,0)
	PD($\boldsymbol{P_1}$)	+	-	Order($\boldsymbol{P_1}$)=2	(*,*,0)
				$\boldsymbol{P_1}$=(*,*,0)	(*,*,0)
				$\boldsymbol{P_1}$=(0,0,0)	(0,0,0)