Error-free protection of EC point multiplication by modular extension

Martin Seysen

February 21, 2017

Giesecke & Devrient GmbH, Prinzregentenstraße 159, D-81677 München,
e-mail: `m.seysen@gmx.de`

Abstract

An implementation of a point multiplication function in an elliptic-curve cryptosystem can be attacked by fault injections in order to reveal the secret multiplier. A special kind of such an attack is the sign-change fault attack. Here the result of a point multiplication is changed in such a way that it is still a point on the curve. A well-known countermeasure against this kind of attack is to perform the point multiplication on a modular extension of the main curve by a small curve. Then the result is checked against the result of the same point multiplication recalculated on the small curve. The problem with this countermeasure is that the point at infinity on the small curve may be reached as an intermediate result with a non-negligible probability. In this case the comparison with the result on the small curve is either faulty or meaningless. We propose a variant of the modular extension countermeasure where the point at infinity is never reached as an intermediate result on the main or on the small curve.

Keywords: elliptic curve, point multiplication, modulus extension

1 Introduction

One of the most fundamental operations in elliptic curve cryptography (ECC) is point multiplication, i.e. the multiplication of a point on the curve with a scalar. This is used in almost all ECC algorithms, such as ECDSA, ECDH and also in pairing-based EC cryptography.

There are several different ways to represent an elliptic curve. The Weierstrass form of a curve (discussed in the next section) is widely used in ECC standards [16, 15] It is well known that the standard formula for point addition fails on a curve in Weierstrass form in certain cases. In this paper we describe an error-free implementation of the ECC point multiplication resistant against side-channel and fault attacks, which is most useful for curves in Weierstrass form.

In a side channel attack, the attacker observes information such as timing differences, power consumption or electromagnetic radiation leaking from an implementation. He may also observe many executions of a cryptographic operation with different inputs, and perform statistical analysis on the input and on the leaking information in order to obtain information about the secret data processed in that implementation. A common countermeasure against side-channel attacks is to add dummy operations when an operation is executed conditionally, so that the attacker cannot see if the condition is satisfied.

In a fault attack, the attacker changes the behavior of an implementation by inducing faults, e.g. by laser attacks, inserting glitches, etc. One type of a fault attack is the safe-error-attack, where the attacker disturbs an operation that may be a dummy operation under a certain condition. That condition may be guessed depending on the effect onto the final result.

In a differential fault attack, the attacker tries to change the value of a certain variable, and to observe the effect on the final result. In ECC, two of the most relevant attacks against point multiplication are the invalid-curve fault attack [5] and the sign-fault change attack [3]. At an invalid-curve fault attack, the attacker tries to change a point on a curve to a point lying on a weaker curve. Then it may be easier to attack point multiplication on the weaker curve. For a sign-fault change we assume that in some places either a point addition or a point subtraction enters into the point multiplication. When changing an addition to a subtraction or vice versa, this has a well-defined effect on the final result that an attacker can use to obtain a bit of the multiplier.

Fan et al. [7] propose a combination of a of fault and a side-channel attack, where a specially prepared point enters into the point multiplication. They assume that this point can be changed to a point of low order on a different curve by a single bit flip and that the occurrence of the neutral element of the group may be detected by side-channel analysis. Goubin [9] proposes a side-channel attack based on a similar idea. He assumes that specific points exist on a curve which can easily be detected by side-channel analysis, e.g. points with one zero co-ordinate.

2 Point multiplication

An elliptic curve group is an algebraic group, and the elements of that group are the points on the projective plane over a field $\mathbb{F}$ satisfying a certain equation of degree 3 or 4. In most cases relevant for cryptography, point multiplication is performed in a subgroup of known prime order of an elliptic curve group over a finite field $\mathbb{F}$.

A point on the curve is usually represented as a pair $(x, y) \in \mathbb{F}^2$. Here $\mathbb{F}^2$ represents the affine plane over $\mathbb{F}$, considered as a subset of the projective plane over $\mathbb{F}$. A variety of different types of equations is used to define an elliptic curve, e.g. the short Weierstrass form, which is $y^2 = x^3 + ax + b$ for fields of characteristic $p > 3$ or $y^2 + xy = x^3 + ax^2 + b$ for fields of characteristic 2. Edwards or Montgomery curves are based on different types of equations.

The neutral element of the group on a curve in Weierstrass form is the point at infinity which cannot be represented in affine co-ordinates.

An ECC point multiplication algorithm is constructed from simpler ECC operations, namely the doubling of a point and the addition or subtraction of two points. When performing these operations on curves in Weierstrass form, the following problems arise. The standard addition formula for computing $P_1 + P_2$, with P_1, P_2 points on the curve, fails in case $P_1 = \pm P_2$. There is a different formula for computing $2P_1$. Of course, all computations in affine co-ordinates fail when the result is the point at infinity. In this paper we assume that point doublings and additions may fail only in the cases mentioned above. There are also unified formulas for point doubling and addition at the cost of some extra field operations, see [4].

Note that additions and doublings in affine coordinates require divisions in $\mathbb{F}$, which are expensive and also vulnerable to side-channel attacks. For division-free practical implementations of point multiplication, projective co-ordinates (x, y, z) are used to represent a point on the curve. Variants of projective co-ordinates, e.g. Jacobian projective co-ordinates, allow even more efficient implementations.

3 Balanced Binary Representation of the Multiplier

We have seen that there is no unified formula for point doubling and addition for elliptic curves in Weierstrass form without additional cost. Thus in a naive implementation of double and add, as in Algorithm 1 for point multiplication, an attacker can easily distinguish between these two operations, so that the multiplier k becomes completely visible for him.

Algorithm 1 *Point multiplication by naive double and add*

```
Input     Point $P$ on curve $E$, integer $k = \sum_{j=0}^{i-1} k_j 2^j, k_j \in \{0,1\}, k_{i-1} = 1$.
Output    $Q = kP$.

[1]  put     $Q = P$
[2]  for     $j$ from $i-2$ down to 0 do :
[3]              $Q = 2 \cdot Q$
[4]              if  $k_j = 1$ then  $Q = Q + P$
[5]  output  $Q$.
```

Coron [6] suggests to perform addition in Step 4 always, and to drop the result, if it is not needed. This opens the door to safe-error attacks as discussed in section 1. There is a variety of other techniques less susceptible to safe-error attacks, see e.g. [10] for an overview. So called *window methods* have been proposed to save additions. Here the idea is to precompute a table of small multiples of the point to be multiplied, and to perform a fixed number of subsequent doublings before a single addition takes place, see e.g. [8, 12] for an overview. The number of subsequent doublings is called the *window size* of the algorithm.

In the remainder of this section we will review the balanced binary point multiplication in [14], which is also a window method, and we will present a variant of that method where the point at infinity of the curve is never reached as an intermediate result. A very similar method has been proposed in [12]. It will turn out in the next section, that the balanced binary point multiplication can easily be protected against sign-fault attacks by a suitable modular extension of the curve.

The following Lemma is easy to show, see [12, 14]:

Lemma 1 *Every odd integer k with $|k| < 2^{ei}$ can be represented as*

$$k = \sum_{j=0}^{i-1} k_j 2^{je} , \quad k_j \textit{ odd}, \; -2^e < k_j < 2^e .$$

So we may present the algorithm for balanced binary multiplication of a point P on an elliptic curve E with an odd multiplier k as follows:

Algorithm 2 *Balanced binary point multiplication*

```
Input     Point $P$ on curve $E$, window size $e$, odd integer $k$ with
          $k = \sum_{j=0}^{i-1} k_j 2^{je}$, $k_j$ odd, $-2^e < k_j < 2^e$.
Output    $Q = kP$.

[1]  put     $T[j] = j \cdot P$ for all odd $j$ with $-2^e < k_j < 2^e$
[2]  put     $Q = T[k_{i-1}]$
[3]  for     $j$ from $i-2$ down to 0 do :
[4]             $Q = 2^e \cdot Q$    // done by repeated doubling
[5]             $Q = Q + T[k_j]$
[6]  output  $Q$.
```

Since point negation is almost trivial for curves in Weierstrass form, it suffices to store the positive multiples of P in Step 1.

In most cryptographic applications we may assume that point multiplication is performed in an elliptic curve group of known prime order q. So in case of an even multiplier k we may simply call Algorithm 2 with input $k - q$ instead of k, as suggested in [14].

We assume that the addition in Step [5] is simply done by the point addition formula, so that it will fail on a curve in Weierstrass form in case $Q = \pm T[k_j]$.

Distinguishing between doubling (in Step [4]) and addition (in Step [5]) by side channel analysis reveals no information about the multiplier or the point. So we may use the fastest doubling or addition formula available for a specific type of curves.

We want to obtain an error-free version of Algorithm 2 for a point P of prime order q. For window size $e > 1$ the error cases are given by:

Lemma 2 *Let P be a point on the curve E of order q, q prime, and let $e > 1$, $q \gg 2^e$, and assume $|k| < 2q$, k odd, in Algorithm 2. Then apart from the obvious case $k = 0 \pmod q$, Algorithm 2 fails only if one of the following two conditions holds:*

1. $q - 2^{e+1} < k < q + 2^{e+1}\ , \quad k = -q \pmod{2^{e+1}}$
2. $-q - 2^{e+1} < k < -q + 2^{e+1}\ , \quad k = q \pmod{2^{e+1}}$

Proof

The algorithm fails only if the neutral element of the group is reached in Step 4 or if $Q = \pm T[k_j]$ in Step 5. Since the group generated by P has odd order, the neutral element cannot be reached by a doubling operation, so the algorithm may fail in Step 5 only. Put $s_{i-1} = k_{i-1}$, $s_j = 2^e \cdot s_{j+1} + k_j$ for $j < i-1$, and put $Q_j = s_j \cdot Q$. Clearly, $s_0 = k$, and the result Q of Step 5 in round j is equal to Q_j, for j running from $i-2$ down to 0. Assume that the algorithm fails and we have $k \neq 0 \pmod q$. Then $s_j - k_j = \pm k_j \pmod q$ holds for at least one $j \le i-2$. Obviously, the sequence $|s_{i-1}|, |s_{i-2}|, \ldots, |s_0|$ contains odd numbers only and is non-decreasing. Thus all $|s_j|$ are $\le |k|$ and hence less than $2q$. Since $q \gg 2^e$ and, asymptotically, $|s_j| \approx 2^e |s_{j-1}| \ge 4|s_{j-1}|$, we may assume $|s_j| < q - 2 \cdot 2^e$ for $j > 0$. This means that $s_j - k_j = \pm k_j \pmod q$ is impossible for $j > 0$, and since $s_0 = k$, we obtain $k - k_0 = \pm k_0 \pmod q$. By assumption $k \neq 0 \pmod q$, so we have $k = 2k_0 \pmod q$. Since k is odd and $|k| < 2q$, we have $k - 2k_0 = \pm q$, with $|k| \approx q \gg |k_0|$. By definition of the k_i, we have $k_0 = k \pmod{2^e}$ and hence $-k = q \pmod{2^{e+1}}$ in case $k > 0$ and $k = q \pmod{2^{e+1}}$ in case $k < 0$. $k - 2k_0 = \pm q$ implies $|q - |k|| \le |2k_0| < 2^{e+1}$.

□

Once the error cases are known, we may easily add a small multiple of the order q to a given multiplier k_0, such that Algorithm 2 runs error free. More specifically, we have:

Lemma 3 *Let P, E, q, $e > 1$ as in Lemma 2 and let $k_0 \in \mathbb{Z}$ with $0 < k_0 < q$, be a multiplier. Then there is an odd k with $kP = k_0P$, $|k| < 2q$ such that Algorithm 2 computes kP error free. A suitable λ with $-2 \le \lambda \le 1$ and $k = k_0 + \lambda q$ can be effectively computed from the lowest three bits of k_0 and q.*

Proof

Put

$$(s_1, k_1) \quad = \quad \begin{cases} (1, k_0) & \text{if } k_0 \text{ is odd} \\ (-1, q - k_0) & \text{if } k_0 \text{ is even} \end{cases}$$

$$\begin{aligned} (s_2, k_2) &= \begin{cases} (s_1, k_1) & \text{if } k_1 \neq -q \pmod 8 \\ (-s_1, 2q - k_1) & \text{if } k_1 = -q \pmod 8 \end{cases} \\ k &= s_2 \cdot k_2 \end{aligned}$$

Obviously, $k_0 = s_i k_i = k \pmod q$, $s_i = \pm 1$, $0 < k_i < iq$ and k_i is odd for $i = 1, 2$. Assume that Algorithm 2 fails for k and $s_2 = 1$. Then $k_2 = -q \pmod 8$ by Lemma 2, part 1. By definition of k_2 we have $k_1 = -q \pmod 8$. Otherwise we would have $k_1 = k_2, k_1 \neq -q \pmod 8$, contradicting $k_2 = -q \pmod 8$. From $k_1 = -q \pmod 8$ we conclude $-q = k_2 = 2q - k_1 \pmod 8$ by definition of k_2. This implies $4q = 0 \pmod 8$, which is impossible for an odd q.

Assuming that Algorithm 2 fails for k and $s_2 = -1$, we obtain a similar contradiction by using Lemma 2, part 2.
$\square$

A secure implementation of Algorithm 2 for an arbitrary multiplier k_0 with $0 < k_0 < q$ should compute $k = k_0 + \lambda q$ with λ obtained from the lowest three bits of k_0 and q by table lookup.

4 Protecting Point multiplication by modular extension

4.1 Sign fault attacks and known countermeasures

Blömer, Otto and Seifert [3] have introduced sign change fault attacks which change the result of a point multiplication in such a way that it is still a point on the curve. This attack applies especially to window methods, such as Algorithm 2 where an addition in Step 5 may be changed to a subtraction or vice versa.

As a countermeasure, it is suggested in [3] to perform the point multiplication on a modular extension of the main curve by a small curve and to check the result against the result of a point multiplication recalculated on the small curve.

The problem with this countermeasure is that the point at infinity on the small curve may me reached as an intermediate result with non-negligible probability. [3] does not specify how addition $P_1 + P_2$ is to be performed if $P_1 = \pm P_2$ holds on the small but not on the main curve. Depending on how this special case is executed on the combined curve and on the small curve, the final comparison between the result on the combined and on the small curve may be either faulty or meaningless in some cases. Specifically, using a small curve of order r may yield considerably more than a fraction of $1/r$ faulty or meaningless cases. See [13] for a detailed discussion of this issue. Here we just remark that an additional check for $P_1 = \pm P_2$ is undesirable regarding performance, and conditional code execution depending on the result of such a check is undesirable regarding side-channel resistance.

Another type of modular extension has been suggested by Baek and Vasyltsov [1]. Here sign-fault attacks are prevented by choosing a different type of equation for the small curve, such that point negation on the small curve is more complicated than on the large curve presented in short Weierstrass form. Then sign faults induced by fault attacks lead to a false result modulo the small curve. However, this leads a performance penalty for the operation on the combined curve, since the combined curve satisfies a more complicated type of equation than the main curve. Also, this countermeasure makes specific assumptions about the representation of the main and the small curve.

Joye [11] uses a small curve in the ring $\mathbb{Z}/r^2\mathbb{Z}$ for protection against fault analysis, leading to about the same security level as a curve in $\mathbb{Z}/r\mathbb{Z}$ in [1].

4.2 Our new countermeasure

Assuming that the point on the main curve has a known prime order (as it is the case in most cryptographic applications), we propose a simple variant of the modular extension countermeasure based on Algorithm 2, where the point at infinity is never reached on the main or the small curve as an intermediate result.

Our new method makes no assumption about the curve, except that there is a formula for point addition $P_1 + P_2$ which is correct in case $P_1 \neq \pm P_2$ and that there is a formula for point doubling which is correct if the doubled point is not the neutral element. This is the case for curves in Weierstrass form, and we may use the fastest algorithms available for doubling or adding.

All we have to do is to extend the main curve E of characteristic p in Algorithm 2 by a small curve E' of characteristic p' coprime to p, such that E' has a cyclic 2-Sylow group and order divisible by 2^{e+1}. We also choose a point P' on E' of maximum order, with the order of P' a multiple of 2^{e+1}.

Algorithm 3 *Error-free point multiplication with modular extension*

```
Input     Point $P$ of order $q$, $q$ prime, on a curve $E$ of chracteristic $p$,
          window size $e > 1$, multiplier $k$ with $0 < k < q$.
          Point $P'$ of order $2^{e+1} \cdot q'$ on a small auxiliary curve $E'$
          of characteristic $p' < p$.
Output    $Q = kP$.

[1]  Put $k_1 = k + \lambda q$ with $\lambda$ as in Lemma 3 such that Algorithm 2
     computes $k_1 P$ error free.
[2]  Let $E^*$ be the curve obtained by Chinese remaindering $E$ and $E'$.
[3]  Let $P^*$ be the point on $E^*$ obtained by Chinese remaindering $P$ and $P'$.
[4]  Compute $Q^* = k_1 \cdot P^*$ on the curve $E^*$ using Algorithm 2.
[5]  Compute $Q' = k_1 \cdot P'$ on the curve $E'$ using Algorithm 2.
[6]  if $Q^* = Q' \pmod{p'}$
          Output $Q := Q^* \bmod p$
     else
          Output Fault attack detected!
```

Proposition 4 *Algorithm 3 computes the point kP without errors in all cases.*

Proof
It suffices to show that the computation of k_1P is error free on the curve E and that the computation of k_1P' is error free on the curve E'. The computation on E is error free by Lemma 3.

Consider the computation on E'. The point P' on E' has even order. In Step 5 of Algorithm 2 we always add and even multiple Q' of P' to an odd multiple $T'[k_j]$ of P', so that the faulty case $T'[k_j] = \pm Q'$ cannot occur. Clearly, the result of Step 5 is an odd multiple of P'. Thus the input Q' into Step 4 of Algorithm 2 is always an odd multiple of P'. Since P' has order divisible by 2^{e+1}, the point at infinity is not reached when Q' is multiplied by 2^e in Step 4. $\square$

Remark
Algorithm 3 is not limited to curves of prime characteristic. We can also work on a curve E in the field $\mathbb{F}_{p^n}$ with, possibly, $p = 2$. Then we may represent $\mathbb{F}_{p^n}$ by $\mathbb{F}_p[x]/R$ for a suitable polynomial $R \in \mathbb{F}_p[x]$ of degree n. We may choose a small curve E' of order divisible by 2^{e+1} over the field $\mathbb{F}_{p^m}$, with $\mathbb{F}_{p^m}$ represented by $\mathbb{F}_p[x]/R'$ for a suitable polynomial $R' \in \mathbb{F}_p[x]$ of degree m. In case $m < n$, the polynomials R and R' are coprime in $\mathbb{F}_p[x]$, so that we can also combine the curves E and E' by Chinese remaindering them in $\mathbb{F}_p[x]/(R \cdot R')$.

4.3 Finding suitable small curves

While finding large curves of prime (or almost prime) order for cryptographic purposes is a challenging task, we can easily find small curves of order $2^{e+1}q'$, q' prime, for, say, $e \leq 5$ and $2^{30} < 2^{e+1}q' < 2^{64}$.

Hasse's theorem states that if E is an elliptic curve over the finite field $\mathbb{F}_{p^n}$, then the number $|E|$ of points on E satisfies $||E| - (p^n + 1)| \leq 2\sqrt{p^n}$. Thus $|E|$ has one of $O(p^{n/2})$ different possible values. Assuming that the group of the curve E is cyclic and has a large prime factor, a random point G on E is a generator with high probability. Using a baby-step giant-step method, the order of G can be found with time and space complexity $O(p^{n/4})$, which is easy in case $p^n \leq 2^{64}$.

Assuming that the orders of the elliptic curves over a finite field are approximately uniform distributed over their feasible interval, the probability that a curve has order $2^{e+1}q'$ for a prime q' is about $2^{-(e+1)}/\log(p^n)$. This means that small curves to be used in Algorithm 3 are easy to find. A simple C program based on these ideas finds a suitable small curve over a field of prime characteristic p and $e = 5$ in a few seconds for a 32-bit p and and in a few hours for a 64-bit p on a standard PC. The Schoof-Elkies-Atkin (SEA) algorithm [2] could be adjusted to speed up the case of a 64-bit p.

In the next section we will show that our modular extension method is quite effective as a protection against a generalized form of sign-fault attacks. So we believe that using a fixed small curve of order about 2^{30} is sufficient for our purposes and that there is no practical need to implement the SEA algorithm for finding a suitable small curve.

4.4 Protection against term-fault attacks

Instead of a sign-fault attack we consider a more powerful kind of attack that we will call a *term-fault* attack. Here the attacker is allowed to change the additive term $T[k_j]$ in Step 5 of Algorithm 2 to any small multiple $T[k'_j]$ stored in the table of multiples computed in Step 1. Changing a few bits of k_j may lead to such an attack. So this kind of attack is as realistic as the 'classical' sign-fault attack in [3]. More generally, term-fault attacks are applicable to any kind of windowing method in ECC point multiplication or modular exponentiation.

Changing $T[k_j]$ to $T[k'_j]$ in round j in Step 5 of Algorithm 2 changes the result Q to $Q + 2^{ej}(T[k'_j] - T[k_j])P$. Since $|T[k'_j] - T[k_j]|$ is even and less than 2^{e+1}, we may say that a term-fault attack changes Q to $Q + \alpha 2^{\epsilon} P$ for some $\epsilon \in \mathbb{N}$ and odd α with $|\alpha| < 2^e$.

We now assume that the attacker may change the term $T[k'_j]$ in two different rounds of Step 5. This changes the result Q to

$$Q + \alpha_1 2^{\epsilon_1} P + \alpha_2 2^{\epsilon_2} P = Q + 2^{\epsilon'}(\alpha_1 + \alpha_2 2^{\epsilon})P \ ,$$

for suitable $\epsilon_1, \epsilon_2, \epsilon', \epsilon > 0$ and odd $|\alpha_1|, |\alpha_2| \leq 2^e$. Modular extension with a curve E' of order $2^{e+1}q'$ will detect this kind of attack, unless the factor of P in the last equation is zero modulo the order q' of E'. Since q' is odd, it suffices to ensure

$$\alpha_2 2^{\epsilon} \not\equiv \pm\alpha_1 \pmod{q'} \quad \text{for all odd } \alpha_1, \alpha_2,\ 0 < \alpha_1, \alpha_2 < 2^e \text{ and } 0 < \epsilon \leq \epsilon_{\max},$$

with $\epsilon_{\max}$ depending on the maximum size of the main curve to be supported. This condition can easily be checked for, say, $\epsilon_{\max} \leq 1000$, which is sufficient for practical ECC implementations. Adding this check to the C program for generating auxiliary curves in the last section leads to a negligible additional overhead.

So in practice our modular extension method achieves a protection against term-fault attacks at up to two different positions.

4.5 Processing a point on the small curve

We recommend to choose a fixed small curve E' (with a cyclic group) of order $2^{e+1}q'$ for prime q' and to precompute a fixed generator point G' of maximum order on that curve. In order to prevent the attacks of type [7, 9], we recommend to use a random multiple $P' = k'G'$, k' odd, $k' \not\equiv 0 \pmod{q'}$ as the input point of the small curve in Algorithm 3. Then an attacker has less control over the point P^* processed in Algorithm 3.

So we have to do two point multiplications on the curve E'. Before starting Algorithm 3 we compute $P' = k'G'$. In Step 5 of Algorithm 3 we will compute $k_1 P' = (k_1 \cdot k')G'$, where $k_1 \cdot k'$ may be reduced modulo $2^{e+1}q'$. So we just need the capability to multiply a fixed point G' with an odd number $\bar{k}$, $0 < \bar{k} < 2^{e+1}q'$.

Since G' has order divisible by 2^{e+1} and $\bar{k}$ is odd, we could use Algorithm 2 with window size up to e for an error free point multiplication on E'. But for this purpose comb methods are much faster. Especially, Algorithm 6 in [8] computes $\bar{k}P'$ error free for odd $\bar{k}$ and $e \geq 1$, since it involves doubling of odd multiples of G' and adding odd to even multiples of G' only. The fixed multiples of G' used in that algorithm can be precomputed.

5 Conclusion

Modular extension of a curve by a small curve during point multiplication is a well-known countermeasure to prevent fault attacks. Previously known modular extension methods have the problem that the point at infinity may be reached on the small curve during operation, or that calculations on the combined curve are more complicated than on the original curve.

We have presented an error-free window method for point multiplication that allows an error-free modular extension, so that the point at infinity is never reached on the main or on the small curve as an intermediate result. In our method there is no need to hide the difference between point doubling and addition to an attacker, and also no need to test for special cases where the doubling or addition formula may fail. Thus the fastest formulas for both operations can be used.

We do not make any specific assumptions about the internal representation of the points on the curve. We just assume that the formulas for point doubling and addition are correct in all cases where the standard doubling and adding formulas for Weierstass curves in affine co-ordinates are correct.

We have introduced term-fault attacks, which are a generalization of the well-known sign-fault attacks, and which are applicable to all kinds of windowing methods in ECC point multiplication or in RSA exponentiation. Our modular extension method is resistant against term-fault attacks in up to two different rounds of the point multiplication algorithm.

Our method can be used for curves over $\mathbb{F}_p$ and $\mathbb{F}_{2^n}$.

References

[1] Y.-J. Baek and I. Vasyltsov. How to prevent dpa and fault attack in a unified way for ecc scalar multiplication: Ring extension method. In E. Dawson and D.S. Wong, editors, *Information Security Practice and Experience (ISPEC 2007)*, volume 4464 of *Lecture Notes in Computer Science*, pages 225–237. Springer, 2006.

[2] I. F. Blake, G. Seroussi, and N. P. Smart. *Elliptic curves in cryptography*, volume 265 of *London Mathematical Society lecture note series.* Cambridge University Press, pub-CUP:adr, 1999.

[3] J. Blömer, M. Otto, and J.P. Seifert. Sign change fault attacks on elliptic curve cryptosystems. In L. Breveglieri, I. Koren, D. Naccache, and J.P. Seifert, editors, *Fault Diagnosis and Tolerance in Cryptography*, volume 4236 of *Lecture Notes in Computer Science*, pages 36–52. Springer, 2006.

[4] E. Brier and M. Joye. Weierstrass elliptic curves and side-channel attacks. In *PKC: International Workshop on Practice and Theory in Public Key Cryptography*. LNCS, 2002.

[5] Ciet and Joye. Elliptic curve cryptosystems in the presence of permanent and transient faults. *IJDCC: Designs, Codes and Cryptography*, 36, 2005.

[6] S. Coron. Resistance against differential power analysis for elliptic curve cryptosystems. In *CHES: International Workshop on Cryptographic Hardware and Embedded Systems, CHES, LNCS*, 1999.

[7] J. Fan, B. Gierlichs, and F. Vercauteren. To infinity and beyond: Combined attack on ECC using points of low order. In B. Preneel and T. Takagi, editors, *CHES*, volume 6917 of *Lecture Notes in Computer Science*, pages 143–159. Springer, 2011.

[8] Min Feng, Bin B. Zhu, Maozhi Xu, and Shipeng Li. Efficient comb elliptic curve multiplication methods resistant to power analysis, April 12 2005.

[9] L. Goubin. A refined power-analysis attack on elliptic curve cryptosystems. In *PKC: International Workshop on Practice and Theory in Public Key Cryptography*. LNCS, 2003.

[10] M. Joye. Elliptic curves and side-channel analysis. In *ST Journal of System Research*, July 21 2003.

[11] M. Joye. Fault-resistant calculations on elliptic curves, 09 2010. EP Patent App. EP 2228716A1, `http://www.google.com/patents/EP2228716A1?cl=en`.

[12] Katsuyuki Okeya and Tsuyoshi Takagi. The width-w NAF method provides small memory and fast elliptic scalar multiplications secure against side channel attacks. *Lecture Notes in Computer Science*, 2612:328–342, 2003.

[13] P. Rauzy, M. Moreau, S. Guilley, and Z. Najm. Using modular extension to provably protect ecc against fault attacks. Cryptology ePrint Archive, Report 2015/882, 2015. `http://eprint.iacr.org/`.

[14] M. Seysen. Computation of a multiple of a group element for cryptographic purposes, 11 2002. W0 Patent App. WO 2002091332A2, `http://www.google.je/patents/WO2002091332A2?cl=en`.

[15] National Institute of Standards U.S. Department of Commerce and Technology. Digital signature standard (dss). FIPS PUB 186-4, `http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf`.

[16] National Institute of Standards U.S. Department of Commerce and Technology. Recommended elliptic curves for federal government use. `http://csrc.nist.gov/groups/ST/toolkit/documents/dss/NISTReCur.pdf`.