Mathematical and Computer Modelling 55 (2012) 256–267

Contents lists available at SciVerse ScienceDirect

Mathematical and Computer Modelling

journal homepage: www.elsevier.com/locate/mcm

Fault detection and a differential fault analysis countermeasure for the Montgomery power ladder in elliptic curve cryptography

Ihor Vasyltsov[a], Gokay Saldamli[b,*]

[a] System LSI, Samsung Electronics, Yongin-city, Gyeonggi-do 449-711, South Korea
[b] Bogazici University, MIS Department, 34342 Bebek, Istanbul, Turkey

ARTICLE INFO

Article history:
Received 25 September 2010
Received in revised form 27 February 2011
Accepted 7 June 2011

Keywords:
Montgomery power ladder
Elliptic curve cryptography
Side-channel attack
Countermeasure

ABSTRACT

We describe a new fault detection method in elliptic curve scalar multiplication deployments using the Montgomery power ladder. An attack based on the arithmetic properties of the Montgomery power ladder algorithm could be avoided by a clearly defined differential fault analysis countermeasure that is extremely efficient against sign-change fault analysis over prime fields. In order to give a complete analysis of the proposed countermeasure, our mathematical models are supported by some software routines implementing various schemes over prime and binary fields. According to our analysis, we report that the performance of the proposed countermeasure meets the theoretical bounds for the checking-at-the-end method, and requires reasonable overhead for the others.

1. Introduction

A reliable cryptographic system starts with correct cryptographic primitive computations. In general, bugs, implementation mistakes, environmental influences, etc., could cause malfunctioning systems/devices that could have a broader negative effect on the larger parent systems. In particular, with the so-called engineering attacks, cryptanalysts focus on the fault influences of specific hardware implementations to collect some faulty output. Later, this information is compared and analyzed with correct results in order to harvest some partial or total compromise of the secret information. Indeed the described approach defines the so-called differential fault analysis (DFA). The early studies [1–6] discuss the applicability of DFA attacks to elliptic curve cryptography (ECC) and advocate the necessity of the implementation of DFA countermeasures in modern ECC signature-generation devices.

By the time simple power analysis (SPA) techniques [7] were introduced, SPA was immediately taken as a serious threat to the payment systems in use. This attracted the interest of significant number of researchers in the security community. However, due to the practical complications in the analysis setup, fault attacks have not been considered as an immediate threat; hence, the research on DFA countermeasures has failed to flourish to a same extent as the side-channel countermeasures. A recent study by Fan et al. [8] confirms this information by exploring the known side-channel and fault attacks and related countermeasures. In fact, the same study points out that there is still much to do in DFA research and in this respect: our study could be considered as a move in this direction. To address the most recent progressive work in the field, we mention [9], which demonstrates a weak-curve-based analysis aiming to replace the strong curves with weaker ones by fault injections. This is an elegant countermeasure for twist curves, but it does not give any protection against DFA. Another work [10] is named as coherence check, which picks a valid pattern and then uses this in verifying the intermediate or final results in scalar multiplication. In their work [11], Baek and Vasyltsov extended Shamir's idea of improving RSA-CRT

* Corresponding author.
E-mail address: gokay.saldamli@boun.edu.tr (G. Saldamli).

0895-7177/$ – see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.mcm.2011.06.017

(Rivest-Shamir-Adleman using Chinese Remainter Theorem) to ECC. Later, Joye [12] reported that some faults are undetected under their countermeasure.

A natural countermeasure against fault detection in ECC is simply checking whether a computed point lies on the curve or not. However, first such an approach is costly, and second, a later result [4] describes that this traditional method is subject to sign-change fault attacks. In this respect, we propose a new DFA countermeasure that is computationally efficient and immune to sign-change fault attacks over a prime field. Our method detects faults in elliptic curve scalar multiplication deployments using the Montgomery power ladder (MPL) algorithm. Attacks based on the arithmetic properties of the Montgomery power ladder algorithm could be avoided efficiently by the proposed DFA countermeasure defined over prime and binary extension fields.

2. ECC and the Montgomery power ladder

Abstractly, a finite field consists of a finite set of objects together with two binary operations – addition and multiplication – that can be performed on pairs of field elements. These binary operations must satisfy certain compatibility properties. There is a finite field containing q field elements if and only if q is a power of a prime number, and in fact for each such q there is precisely one finite field. The finite field containing q elements is denoted by $\mathbb{F}_q$.

For efficient implementation purposes, two types of finite field are of interest: prime finite fields, $\mathbb{F}_p$, with $q = p$, p an odd prime, and binary extension fields, $\mathbb{F}_{2^m}$, thus with $q = 2^m$ for some $m \geq 1$. In fact, there exist many different ways to represent the elements of $\mathbb{F}_p$. A standard representation is given by the elements by the set of integers $\{1, 2, 3, \dots, p\}$, with addition and multiplication modulo p.

An elliptic curve $E(\mathbb{F}_p)$ over $\mathbb{F}_p$ is determined by parameters $a, b \in \mathbb{F}_p$ satisfying $4a^3 + 27b^2 \neq 0$, and consists of the set of solutions or points $P = (x, y)$ for $x, y \in \mathbb{F}_p$ to the equation

$$y^2 \equiv x^3 + ax + b \bmod p \tag{1}$$

together with an extra point O called the point at infinity. One can show that the set of points on $E(\mathbb{F}_p)$ forms a group under the following addition rule (and a similar rule for point doubling). Let $(x_1, y_1) \in E(\mathbb{F}_p)$ and $(x_2, y_2) \in E(\mathbb{F}_p)$ be two points such that $x_1 \neq x_2$. Then $(x_1, y_1) + (x_2, y_2) = (x_3, y_3)$, where

$$\begin{aligned} x_3 &\equiv \lambda^2 - x_1 - x_2, \\ y_3 &\equiv \lambda(x_1 - x_3) - y_1, \end{aligned} \tag{2}$$

with $\lambda \equiv \frac{y_2 - y_1}{x_2 - x_1}$. The group is abelian, and all computation is performed within the finite field $\mathbb{F}_p$.

Similar to the $\mathbb{F}_p$ case, an elliptic curve $E(\mathbb{F}_{2^m})$ over $\mathbb{F}_{2^m}$ is defined by parameters $a, b \in \mathbb{F}_{2^m}$ satisfying $b \neq 0$ and consists of the set of solutions, or points, $P = (x, y)$ for $x, y \in \mathbb{F}_{2^m}$ to the equation

$$y^2 + xy \equiv x^3 + ax^2 + b \tag{3}$$

together with an extra point O called the point at infinity. The set of points on $\mathbb{F}_{2^m}$ forms a group under the following addition rule (and, again, a related doubling rule). Let $(x_1, y_1) \in \mathbb{F}_{2^m}$ and $(x_2, y_2) \in \mathbb{F}_{2^m}$ be two points such that $x_1 \neq x_2$. Then $(x_1, y_1) + (x_2, y_2) = (x_3, y_3)$, where

$$\begin{aligned} x_3 &\equiv \lambda^2 + \lambda + x_1 + x_2 + a, \\ y_3 &\equiv \lambda(x_1 + x_3) + x_3 + y_1, \end{aligned} \tag{4}$$

with $\lambda \equiv \frac{y_2 + y_1}{x_2 + x_1}$.

The security provided by ECC is guaranteed by the difficulty of the discrete logarithm problem in the elliptic curve group. The discrete logarithm problem is the problem of finding the least positive number, k, which satisfies the equation

$$Q = [k]P = \underbrace{P + P + \cdots + P}_{k \text{ times}}, \tag{5}$$

where P and Q are points on the elliptic curve. Naturally, the basic computation – also called point or scalar multiplication – in ECC is finding the kth (additive) power of an element P in the group. This involves quotients of polynomials in the coordinates of the points. That is, it relies completely upon calculations in the underlying field. There are certain simplifications and (sophisticated) tricks to avoid field operations which are particularly onerous [13]. For instance, instead of employing the affine coordinates used in (1) and (3), utilization of projective coordinate representation following the mapping

$$P(x, y) \mapsto P(X, Y, Z), \tag{6}$$

where $X = x$, $Y = y$, and $Z = z$, is a popular choice in ECC realizations. For further details of ECC, we refer to [3,13,6,14].

2.1. Montgomery power ladder

The Montgomery power ladder (MPL) algorithm was originally proposed by Montgomery in 1987 [15]. The algorithm is very popular in asymmetric cryptology realizations, including digital signature generation, because of its resistance to simple power analysis (SPA) attack. Due to space limitations, here, we give only a brief description of the algorithm and refer readers to [15,16] for further details.

Let $\sum_{i=0}^{t-1} k_i 2^i$ be the binary expansion of scalar k in the point multiplication $Q = [k]P$. Initially, we define two series of variables L_j and H_j as follows:

$$L_j = \sum_{i=j}^{t-1} k_i 2^{i-j}, \qquad H_j = L_j + 1. \tag{7}$$

By carefully arranging the terms, it is possible to get

$$\begin{aligned} L_j = 2L_{j+1} + k_j &= L_{j+1} + H_{j+1} + k_j - 1 \\ &= 2H_{j+1} + k_j - 2. \end{aligned} \tag{8}$$

Moreover, putting this into a more closed form gives an iterative process which requires only doubling and additions.

$$(L_j, H_j) = \begin{cases} (2L_{j+1}, L_{j+1} + H_{j+1}) & \text{if } k_j = 0 \\ (2L_{j+1} + H_{j+1}, 2H_{j+1}) & \text{if } k_j = 1. \end{cases}$$

In an ECC setting, these variables could be mapped into elliptic curve points as follows:

$$(L_j, H_j) \mapsto (P_1, P_2), \tag{9}$$

describing a general MPL for ECC scalar multiplication.

Algorithm 1 Montgomery power ladder

Require: $k = (k_{t-1}, \ldots, k_1, k_0)_2$ with $k_{t-1} = 1$ and $P = (x, y)$.
Ensure: $P_1 = [k]P = (x_3, y_3)$.

```
1: $P_1 \leftarrow P$
2: $P_2 \leftarrow 2P$
3: for $i = t - 2$ to 0 do
4:     if $k_i = 1$ then
5:         $P_1 \leftarrow P_1 + P_2; P_2 \leftarrow 2P_2$
6:     else
7:         $P_2 \leftarrow P_1 + P_2; P_1 \leftarrow 2P_1$
8:     end if
9: end for
10: return $P_1 = [k]P = (x_3, y_3)$
```

Note that, in Algorithm 1, the expressions $P_i \leftarrow 2P_i$ and $P_i \leftarrow P_1 + P_2$ for $i = 1, 2$ correspond to elliptic curve point doubling and addition operations, respectively.

2.2. Fast Montgomery power ladder

At first glance, the MPL seems inefficient compared to the standard binary scanning scalar addition in the elliptic curve group as it has to perform both point addition and doubling operations for every iteration. However, López and Dahab [17] proposed an ingenious idea called the fast Montgomery power ladder (FMPL) algorithm and managed to accelerate the MPL by considering not using the Y-coordinate calculation (for projective coordinate representation) in scalar multiplication.

To be more specific, the point addition in binary fields (i.e. (4) in affine coordinates) and respective point doubling operations can be carried out in projective coordinates as follows:

$$\begin{cases} Z_3 = (X_1 Z_2 + X_2 Z_1)^2 \\ X_3 = x Z_3 + (X_1 Z_2)(X_2 Z_1) \end{cases}$$

$$\begin{cases} Z_1 = Z_1^2 X_1^2 \\ X_1 = X_1^4 + b Z_1^4. \end{cases}$$

Similarly, such an idea is applicable to prime fields, and the respective point addition and doubling could be written as follows if (2) is considered in projective space [18].

$$\begin{cases} X_3 = 2(X_1 Z_2 + X_2 Z_1)(X_1 X_2 + a Z_1 Z_2) + 4b Z_1^2 Z_2^2 - x_D (X_1 Z_2 - X_2 Z_1)^2 \\ Z_3 = (X_1 Z_2 - X_2 Z_1)^2 \end{cases} \tag{10}$$

$$\begin{cases} X_1 = (X_1^2 - a Z_1^2)^2 - 8b X_1 Z_1^3 \\ Z_1 = 4 X_1 Z_1 (X_1^2 + a Z_1^2)^2 + 4b Z_1^4. \end{cases}$$

Since the Y-coordinate is not involved in the point addition or doubling computations, one has to recover its value from the computed X and Z coordinates. Such a calculation needs a finite field inversion that could increase the total cost up to 20%. However, it has to be performed only once at the end of scalar multiplication.

3. Fault analysis in ECC

3.1. Fault model

In a DFA attack, an adversary collects some correct and faulty outputs from a cryptographic device by injecting some intentional errors and then analyzes these outputs in order to obtain some information leakage about the cryptographic secrets.

The first report on the applicability of DFA to ECC-based public key cryptosystems was announced in [1], which states that the parameter b in (1) (or (3)) is not involved in the computation of point addition. Later, similar observations were made for other implementations of scalar multiplication algorithms.

The fact that only one of the ECC parameters is used in the scalar multiplication computation (e.g. in (1)), allows us to use a special point $\bar{P} \in \bar{E}(a, \bar{b})$ or $\bar{P} \in \bar{E}(\bar{a}, b)$.

Assume that $\bar{E}$ is a weak elliptic curve, whose order has a small (or smooth) factor r, and that the order of $\bar{P}$ as an element of $\bar{E}$ is equal to r (i.e. $\mathrm{ord}_{\bar{E}}(\bar{P}) = r$). Now, the discrete logarithms $k\bar{P} \mod r$ are computable in $\langle P \rangle$ (i.e. the subgroup of order r generated by $\bar{P}$). Repeating this process for sufficiently many different points $\bar{P}_i$ yields the values of $k \bmod r$ (where $r_i = \mathrm{ord}_{\bar{E}}(\bar{P}))$ from $\bar{Q}_i = k\bar{P}_i$, which soon could be combined by using the Chinese remainder theorem.

Note that fault injections are more useful if they are induced at the beginning of the scalar multiplication. Otherwise, the probability of successful attack drops dramatically, and converges to zero if applied after the scalar multiplication. Ciet and Joye have expand this basic fault analysis model to a more general setting. In their work [2], they consider the possibility of occurrence of transient and permanent faults in the base point as well as domain parameters. Additionally they give detailed guidelines how to use these faults for successful DFA attacks.

3.2. Sign-change fault attack

In their online report [4], Blömer et al. present a successful sign-change fault attack to the NAF-representation of the secret exponent. Later, in a more mature work [19], the same authors show the applicability of their attack to the MPL as well.

The basic idea of the attack is to inject a fault on the sign of the elliptic curve point P in order to extract the faulty inverse point $\bar{P} = -P$. Note that, in this case, the traditional DFA countermeasure which verifies whether the output point is on the curve or not would fail, since the additive inverse lies on the curve, $\bar{P} = (-P) \in E$. The authors showed that a secret key k of length n can be recovered with $O(n2^m l)$ scalar multiplications with probability bigger than $1/2$, where m is the block size and $l = \frac{n}{m} \log 2n$ is the number of injected sign-change faults to perform the attack.

To avoid a sign-change fault attack, Blömer et al. propose computing two scalar multiplications $Q := [k]P_{pl}$ over the curve E_{pl} and $R := [k]P_l$ over a smaller order curve E_l and suggest checking whether $R \equiv Q \mod t$, where l is a small prime. However, such a countermeasure does not seem efficient for real-world cryptographic applications as it would possibly decrease the performance by more than 30%.

As a last remark on these attacks, we note that the applicability of the sign-change attack to binary fields and the FMPL are still open problems.

3.3. Attack on the branching operator

In their study [20], Yen and Kim present a fault attack to the comparison operation, which can be realized in branching. To be more specific, the implementation of "$a \overset{?}{=} b$" can usually be realized by

SUB a, b (or CMP a, b);
JZ (jump if zero);

Notice that these instructions depend on the zero flag, and an attack on the zero flag can cause a successful DFA attack.

4. DFA countermeasure for ECC

4.1. Traditional countermeasures

To avoid DFA for ECC, various countermeasures have been developed [1–4,13]. In most of these proposals, a fault detection (FD) is followed by an action on the detected fault, as seen in Fig. 1.

For instance, the standard DFA countermeasure which can be applied to any of the known cryptographic systems computes the same scalar multiplication twice in two independent calculations, and if the outcome of these calculations

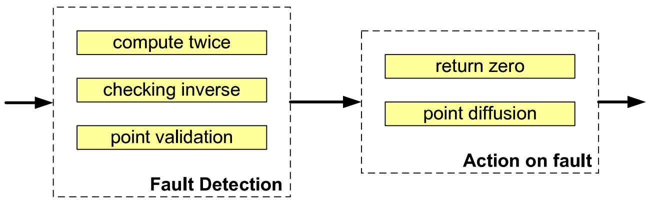

Fig. 1. DFA countermeasure architecture.

is the same, it is assumed that no fault has occurred and the calculated value is returned; otherwise no output is generated. The disadvantage of the basic approach is the duplication of resources (time or area).

Another approach in the FD stage is simply checking the inverse and deciding whether the input of the forward operation is equal to the output of the inverse calculation. Here again, the method is applicable to any cryptographic setting, if the forward and inverse computations are considered as encryption and decryption, respectively. However, the computational overhead is again significant.

Another FD method, particularly applicable to ECC, is checking whether the faulty point $\bar{Q}$ is on the curve or not. The probability of getting a faulty point on E is very low, and the cost of this validation is reasonable. However, such schemes are not immune against sign-change fault attacks as described in the previous section.

On the other hand, the generic action on faults could mislead the attacker. For instance, the zero output action approach clears the sensitive information in the registers and outputs the infinity point in case an error is detected, while a point-diffusion action corrupts the value of the output point, and thus makes the DFA attack infeasible.

4.2. Proposed DFA countermeasure for the MPL

The main idea of the proposed DFA countermeasure relies on the basics of the MPL. In fact, Fisher et al. [5] present a similar countermeasure for RSA implementations. In this study, we prove that this approach is applicable to ECC and then utilize it.

A careful analysis of (7) shows us that in a normal (non-faulty) computation the difference between the temporary variables H_j and L_j is always equal to 1. If the mapping (9) and the initialization of the temporary variables P_1 and P_2 in Algorithm 1 are considered, the difference between the temporary points P_1 and P_2 in the general MPL would always be equal to the difference between points during initialization. In other words, it is equivalent to the base point P. Therefore, we can write

$$H_j - L_j = 1 \mapsto P_{2_j} - P_{1_j} = P. \tag{11}$$

If this relation is stored during the non-faulty computation, we would be sure about the correctness of the computation by simply checking the difference of P_1 and P_2 in any one of the following three equations:

$$P_2 - P_1 \overset{?}{=} P, \qquad P_2 - P \overset{?}{=} P_1 \quad \text{and} \quad P_1 + P \overset{?}{=} P_2. \tag{12}$$

To prove this claim, assume that an attacker injects a fault into the point P_{1_j}, getting corrupted point $\bar{P}_{1_j} = P_{1_j} + P_{\Delta_j}$. Considering the case for $k_j = 0$, the updated values will be computed as

$$\begin{aligned} P_{1_{j+1}} \leftarrow \bar{P}_{1_j} + P_{2_j} &= P_{1_j} + P_{\Delta_j} + P_{2_j} \\ &= P_{1_j} + P_{\Delta_j} + P_{1_j} + P \\ P_{2_{j+1}} \leftarrow 2P_{2_j} = 2(P_{1_j} + P).& \end{aligned}$$

To determine the existence of the fault we simply check whether $P_{2_{j+1}} - P_{1_{j+1}}$ is equal to P or not:

$$\begin{aligned} P_{2_{j+1}} - P_{1_{j+1}} &= (2(P_{1_j} + P)) - (P_{1_j} + P_{\Delta_j} + P_{1_j} + P) \\ &= P - P_{\Delta_j} \neq P. \end{aligned}$$

We conclude that fault detection is possible, and for the next iteration $k_j = 1$ we write

$$\begin{aligned} P_{2_{j+2}} &\leftarrow P_{1_{j+1}} + P_{2_{j+1}} = 4P_{1_j} + P_{\Delta_j} + 3P \\ P_{1_{j+2}} &\leftarrow 2P_{1_{j+1}} = 4P_{1_j} + 2P_{\Delta_j} + 2P. \end{aligned}$$

Checking for $P_{2_{j+2}} - P_{1_{j+2}} = P$, we have

$$P_{2_{j+2}} - P_{1_{j+2}} = P - P_{\Delta_j} \neq P.$$

Therefore, the proposed countermeasure can detect a fault at any iteration (or after) of the scalar multiplication process, as the other cases can be proved in a similar manner. For fault detection, it is possible to use different checking techniques including regular, at-the-end, and random. The difference of these techniques is suggested by their names, giving the timing

Algorithm 2 (At-the-end checking MPL)

```
Require: $k = (k_{t-1}, \dots, k_1, k_0)_2$ with $k_{t-1} = 1$ and $P = (x, y)$.
Ensure: $P_1 = [k]P = (x_3, y_3)$.
1: $P_1 \leftarrow P$
2: $P_2 \leftarrow 2P$
3: for $i = t - 2$ to 0 do
4:     if $k_i = 1$ then
5:         $P_1 \leftarrow P_1 + P_2; P_2 \leftarrow 2P_2$
6:     else
7:         $P_2 \leftarrow P_1 + P_2; P_1 \leftarrow 2P_1$
8:     end if
9: end for
10: if $P_2 - P_1 \neq P$ then
11:     action on fault
12: end if
13: return $P_1 = [k]P = (x_3, y_3)$
```

of the fault-checking process during computation. In this section, we present the checking-at-the-end method, which is the most common and simplest of these. We present the regular and random checking methods in Appendix A for completeness.

Algorithm 2 gives the proposed MPL with checking-at-the-end DFA countermeasure.

This technique performs the checking operation at the end, right after finishing the scalar multiplication. If a fault occur during scalar multiplication, it naturally[1] corrupts the equivalence in (12) and thus the fault will be detected and action on the fault will be performed.

Note that the comparison operation in the proposed algorithm can also be attacked using the approaches that have already been discussed in Section 3.3. However, it is possible to avoid these using the so-called "branchless DFA countermeasure" method proposed by Vasyltsov et al. [21]. The branchless DFA countermeasure for ECC is based on a fault-diffusion technique in which simultaneous fault detection and action on the fault operations are carried. The main idea is simply xoring the following operations, as below[2]:

$$P_1 \leftarrow P_1 \oplus (P_2 - P_1) \oplus P.$$

The fault-diffusion technique is based on the idea of changing the originally computed point P_1 to an unpredictable point P_1'' if any fault is detected. If no fault occurs during the computation, then $(P_2 - P_1) \oplus P = 0$ as a result of the equivalence in (12). Thus, the originally computed point P_1 remains the same. Otherwise, any fault during the scalar multiplication computation will cause an inequivalence in (12) (i.e. $(P_2 - P_1) \oplus P \neq 0$) and the point P_1'' would hardly be predictable provided that the faults are random. The following algorithm gives a modified version of Algorithm 2.

Algorithm 3 (MPL with branchless DFA countermeasure)

```
Require: $k = (k_{t-1}, \dots, k_1, k_0)_2$ with $k_{t-1} = 1$ and $P = (x, y)$.
Ensure: $P_1 = [k]P = (x_3, y_3)$.
1: $P_1 \leftarrow P$
2: $P_2 \leftarrow 2P$
3: for $i = t - 2$ to 0 do
4:     if $k_i = 1$ then
5:         $P_1 \leftarrow P_1 + P_2; P_2 \leftarrow 2P_2$
6:     else
7:         $P_2 \leftarrow P_1 + P_2; P_1 \leftarrow 2P_1$
8:     end if
9: end for
10: return $P_1 \oplus (P_2 - P_1) \oplus P$
```

The computational overhead for the proposed countermeasure is just a single point addition operation which requires ten finite field multiplications. Additionally, it requires just a few more multiplications for the coordinate normalization (for example, two finite field multiplications for the projective representations, or four finite field multiplications for the Jacobian). This is negligible compared to the number of finite field multiplications for the whole scalar multiplication.

[1] We reasonably assume that the attacker might not inject two random faults which can satisfy the checking equation.

[2] Provided that any elliptic curve point is represented by projective coordinates (X, Y, Z), the XOR operation is performed on their coordinates, which in fact, have to be normalized to the common Z-coordinate before the operation.

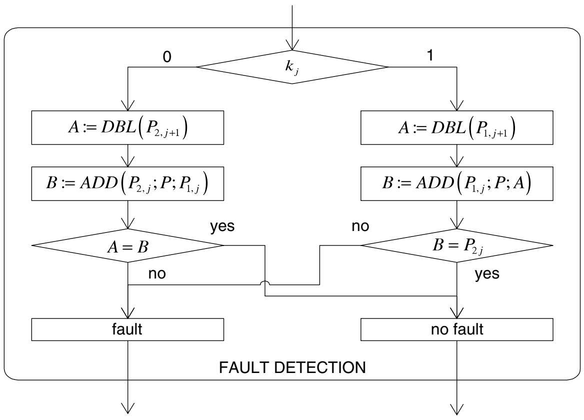

Fig. 2. Computational flow of the proposed fault detection method for the FMPL.

4.3. Fault detection method for the FMPL in ECC

Since the Y-coordinates of the points are not involved in the scalar multiplication computation process, straightforward implementations of the proposed DFA countermeasures for the FMPL are not possible. The most intuitive way to get the Y-coordinates of the points P_1 and P_2 is to perform coordinate conversion, but this conversion requires additional finite field operations (i.e. about ten multiplications and one inversion per Y-coordinate [17,18]). Although this overhead is not critical for the checking-at-the-end method, it may cause significant performance loss for some implementations including the proposed random and regular checking methods.

To solve this problem, we propose to use another method to perform the checking operation in the FMPL case. From (8), the very first description of the MPL, we get

$$\begin{aligned} H_j = 2L_{j+1} + k_j + 1 &= L_{j+1} + H_{j+1} + k_j \\ &= 2H_{j+1} + k_j - 1. \end{aligned} \tag{13}$$

This equation allows us to propose an alternative relation for the checking process:

$$H_j = L_j + 1 \qquad 2H_{j+1} = H_j + 1|_{\text{if } k_j=0},$$

which can be mapped into the elliptic curve point's relations as follows:

$$\begin{aligned} &H_j = L_j + 1 \mapsto P_{2_j} = P_{1_j} + P \\ &2H_{j+1} = H_j + 1|_{\text{if } k_j=0} \mapsto 2P_{2_{j+1}} = P_{2_j} + P|_{\text{if } k_j=0}. \end{aligned}$$

As mentioned in Section 2.2, in order to calculate the X-coordinate of the sum of two points P_1 and P_2 without using the Y-coordinates, the knowledge of the difference of these points $P_D = P_2 - P_1$ is needed. That is why for computing the above equation for $(P_{1_j} + P)$ term what we need to know is the difference point $P_D = (P_{1_j} - P)$, in addition to the part we know. Careful examination of Eq. (13) shows us that

$$L_j - 1 = 2L_{j+1} + k_j + 1|_{\text{if } k_j=1} = 2L_{j+1},$$

and thus, after mapping into the elliptic curve point relations, we have

$$P_D = P_{1_j} - P = 2P_{1_{j+1}}.$$

Hence, to compute $2P_{2_{j+1}}$ (for $k_j = 0$), we need to store the value of $P_{2_{j+1}}$ (from the previous iteration) and to perform the elliptic curve point doubling operation.

With the above mathematical analysis, we introduced two additional variables and developed a fault detection method. To be specific, in the FMPL case, the elliptic curve point addition operation is based on the three input parameters (namely adding point, added point, and difference point). In Fig. 2, the addition operation is marked as $ADD(P_A, P_B, P_C)$, where P_A is the adding point, P_B is the added point, and P_C is the difference point.

Observe from Fig. 2 that the values of temporary points P_1 and P_2 are involved in the fault detection, which shows the ability to detect a fault in any computed point during scalar multiplication.

4.4. Point addition formula for fault detection in the FMPL

Since $X_D = x$, and $Z_D = 1$, the point addition computation in prime fields (10) involves the X-coordinate of the difference point alone. However, this is not correct for the developed fault detection method in the FMPL [22]. Therefore, we outline

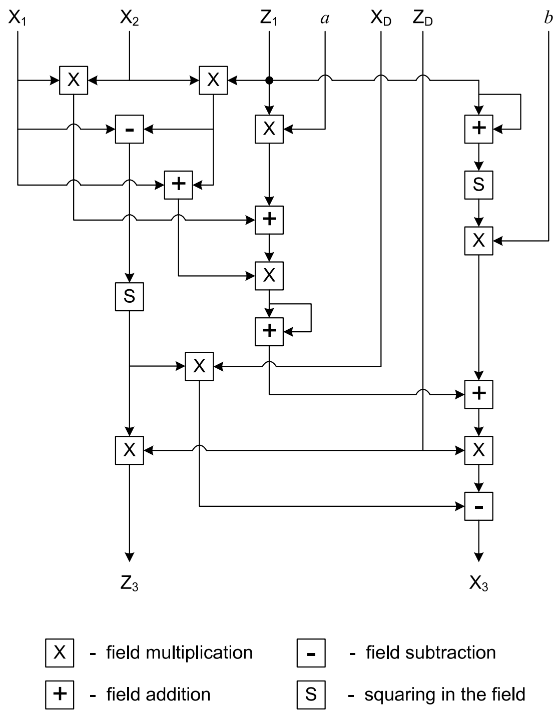

Fig. 3. The flow of the proposed fault detection method for the FMPL in prime fields.

some modifications to the traditional ADD equation as follows. Note that, for the proposed fault detection method, the base point P is originally defined in affine coordinates, so we assume that $Z_2 = 1$.

$$\begin{cases} X_3 = Z_D[2(X_1 + X_2Z_1)(X_1X_2 + aZ_1) + 4bZ_1^2] - X_D(X_1 - X_2Z_1)^2 \\ Z_3 = Z_D(X_1 - X_2Z_1)^2. \end{cases}$$

In Fig. 3, the computational flow to realize the above equation has been illustrated. Note that, for a polynomial basis, the ECC finite field squaring operation can be substituted by multiplications, so the total number of required finite field multiplications is 10.

Similarly, the following adjustment could be done for binary extension fields [23] (assuming that $Z_2 = 1$):

$$\begin{cases} Z_3 = Z_D(X_1 + X_2Z_1)^2 \\ X_3 = X_D(X_1 + X_2Z_1)^2 + Z_DX_1X_2Z_1. \end{cases}$$

Since the finite field squaring operation could be substituted by multiplication and shift operations for polynomial and optimal normal basis (ONB) cases respectively, the total number of required finite field multiplications can easily be calculated from Fig. 4 as 5 for the ONB and 6 for the polynomial basis.

4.5. The FMPL with DFA countermeasure in ECC

The above analysis allows us to introduce a DFA countermeasure for the FMPL in ECC use. In this respect, Algorithm 4 presents an FMPL with checking-at-the-end DFA countermeasure. Notice that to realize Algorithm 4 one would need two additional temporary points, T_1 and T_2. For completeness, we present the FMPL with regular and random checking countermeasures in Appendix B.

5. Security analysis

5.1. Resistance to traditional DFA attack

In order to have a realistic analysis of our proposed DFA countermeasure, we adopt the terms and notions of [24] to our needs. To be informative, the authors mentioned four possible fault attack models, namely precise bit errors (sign-change fault attack is not included), precise byte errors, unknown byte errors, and random errors. It is obvious that the realization of precise bit errors is not trivial and requires an advanced attacker having some costly experimental setting. On the other hand, random error insertion seems easier and more realistic.

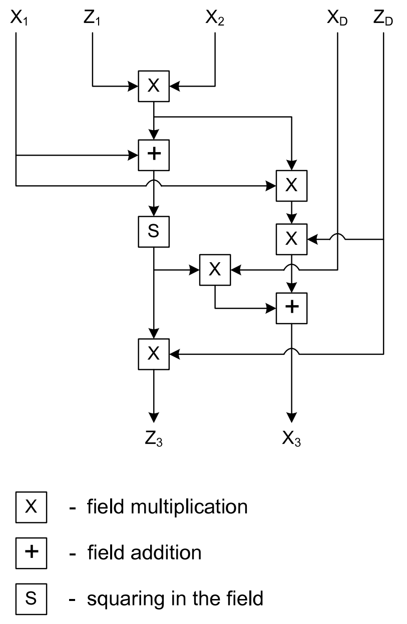

Fig. 4. The flow of the proposed fault detection method for the FMPL in binary fields.

Algorithm 4 (At-the-end branchless FMPL)

```
Require: $k = (k_{t-1}, \dots, k_1, k_0)_2$ with $k_{t-1} = 1$ and $P = (x, y)$.
Ensure: $P_1 = [k]P = (x_3, y_3)$.
1:  $P_1 \leftarrow P$
2:  $P_2 \leftarrow 2P$
3:  for $i = t - 2$ to 0 do
4:      $T_1 \leftarrow P_1$; $T_2 \leftarrow P_2$
5:      if $k_i = 1$ then
6:          $P_2 \leftarrow 2P_2$; $P_1 \leftarrow P_1 + T_2$
7:      else
8:          $P_1 \leftarrow 2P_1$; $P_2 \leftarrow T_1 + P_2$
9:      end if
10: end for
11: if $k_i = 1$ then
12:     $T_1 \leftarrow 2T_1$; $T_2 \leftarrow P_1 + P$
13:     return $P_1 \oplus P_2 \oplus T_2$
14: else
15:     $T_2 \leftarrow 2T_2$; $T_1 \leftarrow P_2 + P$
16:     return $P_1 \oplus T_2 \oplus T_1$
17: end if
```

For the security analysis, we consider the precise byte error model as the most realistic and dangerous fault attack [25,26]. Such a model assumes that attackers could inject single byte errors any time they want but could only determine the location of the faulty byte and not the bits (i.e. the position of the flawed bits on the faulty bytes cannot be settled by the attacker). This means that that new faulty value can only be known with the probability 2^{-8} by the attacker (we assume that faults are uniformly distributed, i.e. the probability of inserting faults is equivalent for every possible value).

Even though injecting a fault into a byte is more realistic, to bypass the proposed DFA countermeasures, the attacker has to keep the equality of (12), which requires modifying both points P_1 and P_2. Since to represent an elliptic curve point one needs the coordinates (X, Y, Z) and (X, Z) in the MPL and the FMPL, respectively, one has to inject faults into six registers for the MPL and four registers for the FMPL. Therefore, the probability of a successful attack can be estimated as $\left(\frac{1}{2^8}\right)^6 = 2^{-48}$ and $\left(\frac{1}{2^8}\right)^4 = 2^{-32}$ for the MPL and the FMPL, respectively. In other words, an attacker has to perform up to 2^{48} and 2^{32} faulty computation for the MPL and the FMPL correspondingly in order to have a successful DFA attack, which can again be considered as infeasible.

5.2. Resistance to sign-change fault attack

As mentioned in Section 3.2, a sign-change fault attack is applicable to MPL-based scalar multiplication in prime fields. Therefore, in this section, we consider the resistance of the proposed DFA countermeasure to this attack. Recall that, in order to bypass the proposed DFA countermeasure, the attacker has to modify more than one point to satisfy the equality $P_2 - P_1 = P$ in calculations. As most realizations of MPL in prime field use Jacobian coordinate representations for implementation of point doubling and mixed Jacobian–affine coordinate representations for point addition, we consider the following equations for realization of these operations [3], where $\lambda = Y_2Z_1^3 - Y_1$:

$$\begin{cases} X_3 = (3X_1^2 + aZ_1^4)^2 - 8X_1Y_1^2 \\ Y_3 = (3X_1^2 + aZ_1^4)(4X_1Y_1^2 - X_3) - 8Y_1^4 \\ Z_3 = 2Y_1Z_1 \end{cases} \tag{14}$$

$$\begin{cases} X_3 = \lambda^2 - (X_2Z_1^2 - X_1)^2(X_1 + X_2Z_1^2) \\ Y_3 = \lambda(X_1(X_2Z_1^2 - X_1)^2 - X_3) - Y_1(X_2Z_1^2 - X_1)^3 \\ Z_3 = \lambda Z_1. \end{cases}$$

A sign-change fault attack inserts the inverse of a faulty point $\bar{P} = -P$ into the curve equation (i.e. $\bar{P}(X, Y, Z) = P(X, -Y, Z)$ in prime fields). Careful examination of the point doubling operation (14) shows that injecting a faulty Y-coordinate corrupts the Z-coordinate of the computed point, i.e. $P_3 \neq \bar{P}_3$.

On the other hand, note that, while computing the X-coordinate in the point addition operation, the term $\lambda = (Y_2Z_1^3 - Y_1)^2$ is involved, and if any one of the Y-coordinates is inverted, it will corrupt the X- and Y-coordinates of the computed point. In the other case, if the attacker inverts both of the Y-coordinates, then the Y-coordinate of the computed point will be corrupted due to the term $-Y_1(X_2Z_1^2 - X_1)^3$, and hence the faulty and correct points will differ, $P_3 \neq \bar{P}_3$.

Note that this would cause a spreading fault diffusion in scalar multiplication using the MPL. Hence, the corrupted output would be detected, as was shown in Section 4.2.

6. Conclusion

We have described a new fault detection method for elliptic curve scalar multiplication deployments using the Montgomery power ladder. The proposed method allows one to build efficient and secure DFA countermeasure for the MPL and FMPL in prime and binary fields. According to our security analysis based on respected attack models, a successful attack to our countermeasure would need more than 2^{48} trials according to the chosen attack model. Such number of trials demands very skillful attackers, Therefore, we claim that the security level of the proposed method is satisfactory as the targeted applications are considered.

In order to implement the DFA countermeasure for the FMPL, special equations to realize point addition in prime and binary fields have been developed. Additionally, to resist potential attacks on the branching operator, a modification of the branchless DFA countermeasure is given for the proposed method. According to our cost analysis, the new countermeasure requires only ten finite field multiplications for the MPL in prime fields. This result corresponds to roughly 5% overhead, and hence, compared to the previous results for the overhead (ranging from 20% to 30%) given above, our method gives the most efficient technique for avoiding sign-change fault attacks.

On the basis of the proposed algorithm, software routines for scalar multiplication in prime and binary fields have been developed. The efficiency analysis on profiling information showed that the performance overhead is reasonable and acceptable to theoretical assumptions. As future work, we aim to combine this countermeasure with DPA (differential power analysis) countermeasures, since in a real-world realization resistance to both DPA and DFA would be required.

Appendix A. Regular and random checking DFA countermeasure for the MPL

In addition to the checking-at-the-end DFA countermeasure presented in Section 4.2, here we present our regular and random checking methods. We start with the regular checking technique, which performs the checking operation in every iteration of the scalar multiplication, as shown below.

The advantage of regular checking comes from the early elimination of the fault and following immediate action on it.

On the other hand, the random checking technique performs the checking operation randomly during the scalar multiplication. To be more specific, the checking operation is performed only when the "check" variable gets the value equal to 1.

Notice that, in this implementation, the performance loss will be less than that in the regular checking technique. Another advantage of the proposed random checking DFA countermeasure method is the generated timing difference. This is useful for increasing the complexity of the power analysis, since it increases the number of trials.

Algorithm 5 (Regular checking MPL)

Require: $k = (k_{t-1}, \dots, k_1, k_0)_2$ with $k_{t-1} = 1$ and $P = (x, y)$.
Ensure: $P_1 = [k]P = (x_3, y_3)$.

```
 1: $P_1 \leftarrow P$
 2: $P_2 \leftarrow 2P$
 3: for $i = t - 2$ to 0 do
 4:     if $k_i = 1$ then
 5:         $P_1 \leftarrow P_1 + P_2$; $P_2 \leftarrow 2P_2$
 6:     else
 7:         $P_2 \leftarrow P_1 + P_2$; $P_1 \leftarrow 2P_1$
 8:     end if
 9:     if $P_2 - P_1 \neq P$ then
10:         action on fault
11:     end if
12: end for
13: return $P_1 = [k]P = (x_3, y_3)$
```

Algorithm 6 (Random checking MPL)

Require: $k = (k_{t-1}, \dots, k_1, k_0)_2$ with $k_{t-1} = 1$ and $P = (x, y)$.
Ensure: $P_1 = [k]P = (x_3, y_3)$.

```
 1: $P_1 \leftarrow P$
 2: $P_2 \leftarrow 2P$
 3: for $i = t - 2$ to 0 do
 4:     if $k_i = 1$ then
 5:         $P_1 \leftarrow P_1 + P_2$; $P_2 \leftarrow 2P_2$
 6:     else
 7:         $P_2 \leftarrow P_1 + P_2$; $P_1 \leftarrow 2P_1$
 8:     end if
 9:     $check \leftarrow rand() \bmod 2$
10:     if $check = 1$ then
11:         if $P_2 - P_1 \neq P$ then
12:             action on fault
13:         end if
14:     end if
15: end for
16: return $P_1 = [k]P = (x_3, y_3)$
```

Algorithm 7 (Regular branchless FMPL)

Require: $k = (k_{t-1}, \dots, k_1, k_0)_2$ with $k_{t-1} = 1$ and $P = (x, y)$.
Ensure: $P_1 = [k]P = (x_3, y_3)$.

```
 1: $P_1 \leftarrow P$
 2: $P_2 \leftarrow 2P$
 3: for $i = t - 2$ to 0 do
 4:     $T_1 \leftarrow P_1$; $T_2 \leftarrow P_2$
 5:     if $k_i = 1$ then
 6:         $P_2 \leftarrow 2P_2$; $P_1 \leftarrow P_1 + T_2$
 7:     else
 8:         $P_1 \leftarrow 2P_1$; $P_2 \leftarrow T_1 + P_2$
 9:     end if
10:     if $k_i = 1$ then
11:         $T_1 \leftarrow 2T_1$; $T_2 \leftarrow P_1 + P$
12:         return $P_1 \oplus P_2 \oplus T_2$
13:     else
14:         $T_2 \leftarrow 2T_2$; $T_1 \leftarrow P_2 + P$
15:         return $P_1 \oplus T_2 \oplus T_1$
16:     end if
17: end for
```

Appendix B. Regular and random checking DFA countermeasure for the FMPL

Similar to the MPL case, regular and random checking methods can be applied to the FMPL DFA countermeasure. The following algorithm outlines the branchless regular checking technique for the FMPL.

Finally, we present the FMPL with random checking DFA countermeasure as follows.

Algorithm 8 (Random branchless FMPL)

Require: $k = (k_{t-1}, \ldots, k_1, k_0)_2$ with $k_{t-1} = 1$ and $P = (x, y)$.
Ensure: $P_1 = [k]P = (x_3, y_3)$.

```
1:  $P_1 \leftarrow P$
2:  $P_2 \leftarrow 2P$
3:  for $i = t - 2$ to 0 do
4:      $T_1 \leftarrow P_1; T_2 \leftarrow P_2$
5:      if $k_i = 1$ then
6:          $P_2 \leftarrow 2P_2; P_1 \leftarrow P_1 + T_2$
7:      else
8:          $P_1 \leftarrow 2P_1; P_2 \leftarrow T_1 + P_2$
9:      end if
10:     $check \leftarrow rand() \bmod 2$
11:     if $check = 1$ then
12:         if $k_i = 1$ then
13:             $T_1 \leftarrow 2T_1; T_2 \leftarrow P_1 + P$
14:             return $P_1 \oplus P_2 \oplus T_2$
15:         else
16:             $T_2 \leftarrow 2T_2; T_1 \leftarrow P_2 + P$
17:             return $P_1 \oplus T_2 \oplus T_1$
18:         end if
19:     end if
20: end for
```

References

[1] I. Biehl, B. Meyer, V. Müller, Differential fault attacks on elliptic curve cryptosystems, in: Advances in Cryptology CRYPTO'00, Santa Barbara, CA, USA, pp. 131–146.
[2] M. Ciet, M. Joye, Elliptic curve cryptosystems in the presence of permanent and transient faults, Designs, Codes and Cryptography 36 (2005) 33–43.
[3] D. Hankerson, A. Menezes, S. Vanstone, Guide to Elliptic Curve Cryptography, Springer-Verlag, 2003.
[4] J. Blömer, M. Otto, J.-P. Seifert, Sign change fault attacks on elliptic curve cryptosystems, Cryptology ePrint Archive, Report 2004/227, IACR, 2004. Available online at: http://eprint.iacr.org/2004/227.pdf.
[5] C. Giraud, An RSA implementation resistant to fault attacks and to simple power analysis, IEEE Transactions on Computers 55 (2006) 1116–1120.
[6] I.F. Blake, G. Seroussi, N.P. Smart, Advances in Elliptic Curves in Cryptography, Cambridge University Press, 2004.
[7] P. Kocher, Timing attacks on implementations of Diffie–Hellman, RSA, DSS, and other systems, in: Advances in Cryptology CRYPTO'96, Santa Barbara, CA, USA, pp. 104–113.
[8] J. Fan, X. Guo, E. De Mulder, P. Schaumont, B. Preneel, I. Verbauwhede, State-of-the-art of secure ECC implementations: a survey on known side-channel attacks and countermeasures, in: Hardware-Oriented Security and Trust, HOST, IEEE International Symposium on, 2010, pp. 76–87.
[9] P. Fouque, R. Lercier, D. Real, F. Valette, Fault attack on elliptic curve Montgomery ladder implementation, in: Fifth International Workshop on Fault Diagnosis and Tolerance in Cryptography—FDTC'08, pp. 92–98.
[10] A. Dominguez-Oviedo, On fault-based attacks and countermeasures for elliptic curve cryptosystems, Ph.D. Thesis, University of Waterloo, 2008.
[11] Y.-J. Baek, I. Vasyltsov, How to prevent DPA and fault attack in a unified way for ECC scalar multiplication—ring extension method, in: Information Security Practice and Experience, ISPEC2007, in: LNCS, vol. 4464, pp. 225–237.
[12] M. Joye, On the security of a unified countermeasure, in: Fifth International Workshop on Fault Diagnosis and Tolerance in Cryptography—FDTC'08, pp. 87–91.
[13] I.F. Blake, G. Seroussi, N.P. Smart, Elliptic Curves in Cryptography, Cambridge University Press, 1999.
[14] IEEE, P1363: Standard specifications for public-key cryptography, Draft Version 13, 1999.
[15] P.L. Montgomery, Speeding the Pollard and elliptic curve methods of factorization, Mathematics of Computation 48 (1987) 243–264.
[16] M. Joye, S.-M. Yen, The Montgomery powering ladder, in: CHES 2002, in: LNCS, Springer-Verlag, 2002, pp. 291–302.
[17] J. López, R. Dahab, Fast multiplication on elliptic curves over GF(2^m) without pre-computation, in: Proceedings of International Workshop on Cryptographic Hardware and Embedded Systems, CHES'99, pp. 316–327.
[18] W. Fisher, C. Giraud, E.W. Knudsen, J.-P. Seifert, Parallel scalar multiplication on general elliptic curves over F_p hedged against non-different side-channel attacks, Cryptology ePrint Archive, Report 2002/7, IACR, 2002. Available online at: http://eprint.iacr.org/2002/007.pdf.
[19] J. Blömer, M. Otto, J.-P. Seifert, Sign change fault attacks on elliptic curve cryptosystems, in: FDTC 2006, in: LNCS, vol. 4236, 2006, pp. 36–52.
[20] S.-M. Yen, D. Kim, Cryptanalysis of two protocols for RSA with CRT based on fault infection, in: FDTC 2006, in: LNCS, vol. 4236, 2006, pp. 53–61.
[21] I. Vasyltsov, Y.-J. Baek, H.-K. Son, Branchless DFA countermeasure method for ECC, in: Pre-Proceedings of 6-th International Workshop on Information Security Applications WISA 2005, Jeju Island, S. Korea, pp. 429–439.
[22] I. Vasyltsov, J.-H. Hwang, Apparatus for performing a fault detection operation and method thereof, US Patent Application 2008003144, 2008.
[23] I. Vasyltsov, Apparatus for performing a fault detection operation and method thereof, US Patent Application, 20080031444, 2008.
[24] M. Ciet, M. Joye, Practical fault countermeasure for Chinese remaindering based RSA, Presented at FDTC'05, 2005.
[25] J. Blömer, M. Otto, J.-P. Seifert, A new CRT-RSA algorithm secure against Bellcore attacks, in: CCS 2003, ACM SIGSAC, ACM Press, 2003, pp. 311–320.
[26] S.P. Skorobogatov, R.J. Anderson, Optical fault induction attacks, in: 4th International Workshop on Cryptographic Hardware and Embedded Systems, CHES'02, Springer-Verlag, 2003, pp. 2–12.