Low-cost fault detection method for ECC using Montgomery Powering Ladder

Duško Karaklajić, Junfeng Fan, Jörn-Marc Schmidt* and Ingrid Verbauwhede
Katholieke Universiteit Leuven, ESAT/SCD-COSIC and IBBT
Kasteelpark Arenberg 10, B-3001 Leuven-Heverlee, Belgium
Email: firstname.lastname@esat.kuleuven.be
*Institute for Applied Information Processing and Communications (IAIK),
Graz University of Technology, Inffeldgasse 16a, 8010 Graz, Austria
Email: Joern-Marc.Schmidt,@iaik.tugraz.at

***Abstract*—When using Elliptic Curve Cryptography (ECC) in constrained embedded devices such as RFID tags, López-Dahab's method along with the Montgomery powering ladder is considered as the most suitable method. It uses x-coordinate only for point representation, and meanwhile offers intrinsic protection against simple power analysis.**

This paper proposes a low-cost fault detection mechanism for Elliptic Curve Scalar Multiplication (ECSM) using the López-Dahab algorithm. Introducing minimal changes to the last round of the algorithm, we make it capable of detecting faults with a very high probability. In addition, by reusing the existing resources, we significantly reduce both performance losses and area overhead compared to other methods in this scenario. This method is suitable especially for constrained devices.

***Index Terms*—Elliptic Curve Cryptosystems (ECC), Montgomery Powering Ladder, Fault Attacks, Low Overhead, López-Dahab algorithm.**

I. Introduction

The implementation of Public Key Cryptography on very constrained devices such as passive RFID tags has always been challenging. Cryptographic systems based on Elliptic Curves (EC) are considered a better choice than RSA on such platforms, since they offer equivalent security at a much smaller key size. A shorter key size implies lower hardware costs and a reduced power consumption. Thus, lightweight implementations of ECC are a hot research topic nowadays. Many clever implementation tricks have been proposed to reduce the computational complexity and hardware usage. For example, using the López-Dahab algorithm [16] together with the Montgomery Powering Ladder (MPL) enables usage of just one coordinate during scalar multiplications, which saves both computation time and storage in hardware.

Implementations of a mathematically secure algorithm might be vulnerable to physical attacks, i.e. attacks that make use of physical information leakage. The two most prominent attacks in this area are side-channel analysis (SCA) and fault analysis (FA). While SCA reveals sensitive data by monitoring information that leaks from side-channels, such as processing time, power consumption, or electromagnetic radiation, FA exploits actively induced faults in the computation. Hence, countermeasures that prevent implementations attacks are vital. Since incorporating countermeasures adds extra computation time and chip area, implementing secure algorithms has been a big challenge for constrained devices such as passive RFID tags. In this paper, we present a low-cost method to protect ECC implementations against fault analysis. We target implementations using López-Dahab algorithm and Montgomery Powering Ladder, which are considered the best combination for constrained devices [20], [12]. The proposed method detects faults induced during the point multiplication with a high probability. It is designed in a way that it (1) has a minimum overhead in terms of computation time and that it (2) reuses existing functions or circuits to reduce the area overhead.

The remainder of this paper is organized as follows: In Section II, we recap basic notations of ECC, give a brief introduction to the considered implementation and describe common fault attacks, before we explain our method in III. Its performance and security is evaluated in Section V and Section IV, respectively. Conclusions are drawn in Section VI.

978-3-9810801-7-9/DATE11/©2011 EDAA

II. ECC and related attacks

A. Elliptic Curve Cryptography

This section gives a brief introduction to ECC. A more comprehensive explanation can be found in [3], [1]. An elliptic curve E over a field $\mathbf{F}$ is defined by the Weierstrass equation:

$$E : y2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \qquad (1)$$
$$a_1, \ldots, a_6 \in \mathbf{F}.$$

For a binary field $\mathbf{F}$, (1) can be simplified to:

$$E_b : y^2 + xy = x^3 + ax^2 + b. \qquad (2)$$

The set of points $(x, y) \in \mathbf{F}^2$ fulfilling (1) together with the point at infinity $\mathcal{O}$ form an additive abelian group denoted as $E(\mathbf{F})$. The point $\mathcal{O}$ is the neutral element of the group. The

main group operations are point addition and point doubling. A group operation on an elliptic curve consists of many operations in the underlying field.

Given a point P and an integer k, computing $Q = k \cdot P$ is called scalar multiplication with scalar k. In most cryptosystems, the scalar corresponds to the secret key. The security of ECC is based on the hardness of the so-called Elliptic Curve Discrete Logarithm Problem (ECDLP), namely, finding a k for two given points P and Q, such that $Q = k \cdot P$. The group given by a curve E can be represented in affine form, i.e. each point is represented by two coordinates: $P = (x_P, y_P)$. An equivalent representation given by projective Jacobian coordinates. In Jacobian coordinates, the projective point $(X, Y, Z) \in \mathbf{F}^3$, $Z \neq 0$ corresponds to the affine point $(x, y) = (X/Z^2, Y/Z^3) \in \mathbf{F}^2$. The point at infinity is represented by $(0, 1, 0)$.

López and Dahab [16] proposed a fast and efficient algorithm for point multiplication on EC. Let $P = (x_P, y_P)$, $Q_1 = (x_1, y_1)$, and $Q_2 = (x_2, y_2)$ be three points on the curve E_b with $Q_2 = Q_1 + P$. Let $Q_3 = Q_1 + Q_2$ and $Q_4 = 2 \cdot Q_i$ for $i \in \{1, 2\}$, then the x-coordinate of Q_3 and Q_4 can be computed without using the y-coordinate of Q_1 or Q_2.

$$\begin{array}{ll} x_3 & = x_P + (\frac{x_1}{x_1+x_2})^2 + \frac{x_1}{x_1+x_2} \\ x_4 & = x_i^2 + \frac{b}{x_i^2} \end{array} \tag{3}$$

A projective version of this algorithm was also proposed. Let $P = (x_P, y_P)$, $Q_1 = (X_1, Z_1)$ and $Q_2 = (X_2, Z_2)$, where $Q_2 = Q_1 + P$ and $x_i = X_i/Z_i$ for $i \in \{1, 2\}$. If $Q_3 = Q_1 + Q_2$ and $Q_4 = 2 \cdot Q_i$, then $Q_3 = (X_3, Z_3)$ and $Q_4 = (X_4, Z_4)$ can be computed as follows:

$$\left\{ \begin{array}{lcl} Z_3 & = & (X_1 Z_2 + X_2 Z_1)^2 \\ X_3 & = & x_P Z_3 + X_1 Z_2 X_2 Z_1, \end{array} \right. \tag{4}$$

$$\left\{ \begin{array}{lcl} X_4 & = & X_i^4 + b Z_i^4 \\ Z_4 & = & Z_i^2 X_i^2. \end{array} \right. \tag{5}$$

When used in the MPL, the López-Dahab formulas have a big advantage over those using three coordinates. This is because throughout the scalar multiplication the precondition $S = Q + P$ for using the formulas is given. The MPL is depicted in Algorithm 1. The Algorithm returns (X_Q, Z_Q), from which $x_Q = X_Q/Z_Q$ can be derived with one division. The recovery of y_Q requires $P = (x_P, y_P), x_Q$ and x_s [16]:

$$y_Q \leftarrow (x_Q + x_P)\{(x_Q + x_P)(x_S + x_P) + x_P^2 + y_P\}/x_P + y_P.$$

While the recovery of the y-coordinate from the result of the López-Dahab scalar multiplication is possible, it is not required for common ECC applications like ECIES (EC Integrated Encryption Scheme), ECDSA (EC Digital Signature Algorithm) or ECDH (EC Diffie-Hellman) [13], [18]. Those schemes use only the x-coordinate of a point, neglecting the y value. Thus, the optimizations of López and Dahab can save all storage space for y-coordinates and the efforts of computing them, resulting in higher efficiency than other proposals. Thus, we aim at protecting the MPL without recovering the y-coordinate using the already implemented circuits.

Algorithm 1 Montgomery Powering Ladder using López-Dahab Coordinates

Require: An integer $k \geq 0$ and a base point $P(x)$
Ensure: $Q(x) = kP$.
1: Set $\leftarrow (k_{l-1} \ldots k_1 k_0)_2$.
2: $X_Q \leftarrow x,\ Z_Q \leftarrow 1,\ X_S \leftarrow x^4 + b,\ Z_S \leftarrow x^2$.
3: **for** i **from** $l - 2$ **downto** 0 **do**
4: **if** $k_i = 1$ **then**
5: $Q \leftarrow Q + S,\ S \leftarrow 2S$.
6: **else**
7: $S \leftarrow Q + S,\ Q \leftarrow 2Q$.
8: **end if**
9: **end for**
10: **Return** Q.

Generally, a computation on an elliptic curve can be performed in affine as well as in projective coordinates. What makes projective coordinates the common choice is that they trade a field inversion in each point operation for an additional coordinate. However, before outputting the result, it is advisable to convert the point back into the affine representation. This not only saves costs of transferring a coordinate, but also prevents an adversary from gaining side-channel information from the projection [17]. Hence, we assume that the final result of the EC Scalar Multiplication (ECSM) is always output in affine coordinates.

B. Fault Analysis on ECC

Adversaries with physical access to a device can actively disturb the cryptographic computation and use the erroneous output (or not even the output, but the reaction of the disturbed device) to derive the secret. In order to do so, the adversary needs to induce faults on the target device. Various methods can be used, such as changing memory bits with a laser or violating setup times by inducing glitches in the clock signal. Readers who are interested in these methods are referred to [15].

Fault attacks can be grouped into three categories, namely, safe-error based analysis [19], [14], weak-curve based analysis [2], [4], [9] and differential fault analysis (DFA) [2]. There is a fundamental difference between these three attack methods and thus their countermeasures. Safe-error attacks are based on the observation that some errors do not change the results, e.g. if dummy operations are tampered. Weak curve attacks try to move a scalar multiplication from a cryptographic strong curve to a $weaker$ curve. Differential fault attacks exploit the difference between a correct output and an erroneous one to retrieve the scalar in small chunks. We give a brief instruction to DFA, since our countermeasure prevents this attack.

Biehl et al. [2] reported the first DFA on an ECSM implementation. For simplicity reasons, we use a right-to-left double and add algorithm to describe this attack. Nevertheless, the same technique can be applied to other versions of the double and add algorithm as well as the Montgomery ladder. Q_i and R_i denote the value of Q and R at the end of the i^{th} iteration,

Algorithm 2 Right-To-Left (upwards) binary double and add algorithm for scalar multiplication

Input: $P \in E(\mathbb{F})$ and integer $k = \sum_{i=0}^{l-1} k_i 2^i$.
Output: $k \cdot P$.

1: $R \leftarrow P$, $Q \leftarrow \mathcal{O}$.
2: **for** $i = 0$ to $l - 1$ **do**
3: If $k_i = 1$ then $Q \leftarrow Q + R$.
4: $R \leftarrow 2R$;
5: **end for**
Return R.

respectively. Let $k(i) = k$ div 2^i and Q'_i be the value of Q_i after a fault has been induced. The attack reveals k from the Most Significant Bits (MSB) to the Least Significant Bits (LSB).

1) Run ECSM once and collect the correct result (Q_n).
2) Run the ECSM again and induce a single bit-flip in Q_i, with $l - m \leq i < l$. We assume that m is small.
3) Note that Q_n=Q_i+$k(i)2^i \cdot P$ and Q'_n=Q'_i+$k(i)2^i \cdot P$. The adversary then tries all possible $k(i) \in \{0, 1, .., 2^m - 1\}$ to generate Q_i and Q'_i. The correct value of $k(i)$ will result in a $\{Q_i$,$Q'_i\}$ that have only one-bit difference.

The attack works in the same way for the left-to-right multiplication algorithm. It also can be applied if k is encoded with any other deterministic code, such as Non-Adjacent-Form (NAF) and w-NAF. It is also claimed that a fault induced at a random moment during an ECSM is sufficient [2].

In [8] Fan et al. provided a systematic overview of implementation attacks and countermeasures for ECC. Here we are focusing on ECSM using Montgomery Powering Ladder without employing y-coordinate. Table I depicts possible attacks and the relevancy of different countermeasures for this algorithm.

TABLE I
ATTACKS AND COUNTERMEASURES ON ECC USING MPL WITHOUT y-COORDINATE

	Fault Attacks						
	Safe- Error		Weak Curve			Differential	
	M Safe-Error	C Safe-Error	Invalid Point	Invalid Curve	Twist Curve	Sign Change	Differential
MPL	√	√	–	–	**H** [10]	√	–

√: effective countermeasure
-: ineffective countermeasure
H: countermeasure helps an attack

Table I shows that MPL without y-coordinate provides inherent protection against safe error and sign change attacks. However, in order to prevent other attacks, additional countermeasures must be implemented. For example, checking the integrity of the curve parameters and the base point coordinates can prevent invalid curve and invalid point attacks, while choosing an elliptic curve which has only strong twists will prevent a twist curve attack. In order to prevent differential fault analysis, coherence of the intermediate and final result should be checked. Another possible method is to blind the scalar k such that the adversary can not gain more bits of k in repeated executions.

III. The Low-Cost Fault Detection Method

In this section we present the proposed implementation of the coherency check (CC) performed at the end of the MPL.

A CC verifies the intermediate or final results to detect faults injected during the computation, in our case the ECSM [5]. If an ECSM uses the MPL, the fact that the difference between S and Q is always P (see Algorithm 1) can be used as the required coherency. CC is an efficient countermeasure against differential fault analysis. However, if not implemented carefully, it introduces significant performance and hardware overhead.

In the case of the Montgomery ladder, a CC evaluates the equation $S = Q + P$ to check the coherency [5], [6]. Note that the addition $Q + P$ can not be performed using López-Dahab algorithm since the difference between two points is not known. Hence, the y coordinates of both Q and P must be recovered. Since the recovery requires finite field inversions, the performance overhead is significant. Besides, the y coordinate of P occupies an extra location in the non-volatile memory. In addition, a straight-forward implementation of the coherency check requires a regular point addition algorithm, which introduces the overhead in both datapath and control part. In this paper, we present an efficient method to perform the coherency check using functions that are already implemented for ECSM.

Let $k = (k_{l-1} \dots k_1 k_0)_2$ be an l-bit scalar and P the base point. The points Q and S denote the intermediate results of the second-last round of the Algorithm 1, which calculates the scalar multiplication $k \cdot P$. If (Q_1, S_1) represents the result of the last round for $k_0 = 1$ and (Q_0, S_0) the result of the last round for $k_0 = 0$, the following equations hold:

$$Q_0 = 2Q; \qquad S_0 = Q + S = 2Q + P,$$

$$Q_1 = Q + S = 2Q + P = S_0; \qquad S_1 = 2S = 2Q + 2P.$$

For fault-free computations, it follows that:

$$S_1 = Q_1 + P. \tag{6}$$

In order to check the coherence (6) between S_1 and Q_1, the last round of Algorithm 1 is modified such that it calculates the results for both possible values of k_0. Algorithm 3 gives the implementation of the proposed countermeasure. Note that the point addition in Line 17 uses the López-Dahab point addition algorithm, since the difference between Q_1 and P is Q_0, which is already calculated (Line 16). Indeed, the affine x-coordinate of Q_0, $x(Q_0)$, is required for that purpose. In case the result is always output in affine coordinates (see Section II), it has to be computed anyway if the last bit $k_0 = 0$. However, if $k_0 = 1$ the computation of $x(Q_0)$ is redundant.

Algorithm 3 Montgomery Powering Ladder with the Coherency Check

Require: An integer $k \geq 0$ and a base point $P(x_P, 1)$
Ensure: $Q(x) = k \cdot P$.
1: Set $\leftarrow (k_{l-1} \dots k_1 k_0)_2$.
2: $X_Q \leftarrow x_P,\ Z_Q \leftarrow 1,\ X_S \leftarrow x_P^4 + b,\ Z_S \leftarrow x_P^2$.
3: **for** i **from** $l-2$ **downto** 1 **do**
4: **if** $k_i = 1$ **then**
5: $\mathbf{Z_Q} \leftarrow (X_Q Z_S + X_S Z_Q)^2$, $X_Q \leftarrow x_P \mathbf{Z_Q} + X_Q Z_S X_S Z_Q$;
6: $\mathbf{Z_S} \leftarrow Z_S^2 X_S^2$, $\mathbf{X_S} \leftarrow X_S^4 + b Z_S^4$;
7: **else**
8: $\mathbf{Z_Q} \leftarrow Z_Q^2 X_Q^2$, $\mathbf{X_Q} \leftarrow X_Q^4 + b Z_Q^4$;
9: $\mathbf{Z_S} \leftarrow (X_Q Z_S + X_S Z_Q)^2$, $\mathbf{X_S} \leftarrow x_P \mathbf{Z_S} + X_Q Z_S X_S Z_Q$;
10: **end if**
11: **end for**
12: $Z_{Q_1} \leftarrow (X_Q Z_S + X_S Z_Q)^2$, $X_{Q_1} \leftarrow x_P Z_{Q_1} + X_Q Z_S X_S Z_Q$;
$Z_{S_1} \leftarrow Z_S^2 X_S^2$, $X_{S_1} \leftarrow X_S^4 + b Z_S^4$;
13: $Z_{Q_0} \leftarrow Z_Q^2 X_Q^2$, $X_{Q_0} \leftarrow X_Q^4 + b Z_Q^4$;
$Z_{S_0} \leftarrow Z_{Q_1}$, $X_{S_0} \leftarrow X_{Q_1}$;
14: $Z_{inv} = \frac{1}{Z_{Q_0} Z_{Q_1}}$;
15: $x(Q_1) = Z_{inv} X_{Q_1} Z_{Q_0}$;
16: $x(Q_0) = Z_{inv} X_{Q_0} Z_{Q_1}$;
17: $Z_T \leftarrow (X_{Q_1} + x_P Z_{Q_1})^2$, $X_T \leftarrow x_{Q_0} Z_T + X_{Q_1} x_P Z_{Q_1}$;
18: **if** $X_{S_1} Z_T \neq X_T Z_{S_1}$ **then**
19: **Return**(Error).
20: **else if** $k_0 = 1$ **then**
21: **Return**(x_{Q_1}).
22: **else**
23: **Return**(x_{Q_0}).
24: **end if**

The variables in Algorithm 3 are presented in projective coordinate system using X and Z coordinates. Thus, we can write the following expressions for the points used in Line 11:

$$Q_1 = (X_{Q_1}, Z_{Q_1});\ \ Q_0 = (X_{Q_0}, Z_{Q_0}).$$

Since one out of Q_0 and Q_1 is the final result, we convert both to an affine coordinate system using the following set of equations:

$$\begin{aligned} Z_{inv} &= \tfrac{1}{Z_{Q_0} Z_{Q_1}}, \\ x(Q_1) &= Z_{inv} X_{Q_1} Z_{Q_0}, \\ x(Q_0) &= Z_{inv} X_{Q_0} Z_{Q_1}. \end{aligned} \tag{7}$$

Equations (7) show that the conversion from projective to affine coordinates of both possible results can be implemented using just one inversion and five field multiplications. Knowing $x(Q_0)$, we can reuse López-Dahab algorithm to add Q_1 and P.

The final step is the comparison of S_1 and $Q_1 + P$. If $T(X_T, Z_T)$ denotes the sum $T = Q_1 + P$ and S_1 denotes point computed in Algorithm 3, the comparison between T and S_1 can be performed evaluating the following equation:

$$X_{S_1} Z_T = X_T Z_{S_1}. \tag{8}$$

If (8) holds, the result is output depending on the last bit k_0. Note that the affine coordinates of both possible outputs are already calculated in (7).

IV. Security Evaluation

The proposed countermeasure detects faults injected during the MPL. This section evaluates security of the proposed implementation by calculating the probability that an injected fault is not detected.

The proposed countermeasure fails to detect a faulty base point provided as an input to MPL. A couple of attacks on EC implementations rely on supplying a base point that moves the whole computation from a cryptographically strong curve to a weak one. A similar attack is possible by injecting a fault right after a sanity check of the point. One example is the twist-based attack when only x-coordinate is used [11]. To prevent such kind of attacks, it is advisable to choose a curve that has strong twists. A more general method to prevent this attack is to check that the result, $Q = k \cdot P$, belongs to the specified curve. In the following analysis we assume this method is always used.

We assume the adversary only induces faults once. When the fault is induced, one or more registers (i.e. X_Q, Z_Q, X_S, Z_S) can be affected. Faults that are induced during a point addition or point doubling are equivalent to faults on the results of that iteration. Thus, we simply treat the faults as forced value changes of X_Q, Z_Q, X_S or Z_S in the end of a specific iteration.

We analyze the probability of a random fault passing the check for an elliptic curve E with a cofactor 2. The same analysis can be applied to curves with different cofactor. Let $\#E$ be the order of the curve, and q be the order of the base point P. Since E has a cofactor 2, we have $\#E = 2q$.

Consider the point addition and doubling function (Eq. 3) as a map:

$$f_{k_i}: \begin{array}{ccc} \mathbf{F} \times \mathbf{F} & \xrightarrow{k_i \in \{0,1\}} & \mathbf{F} \times \mathbf{F} \\ \{x_1, x_2\} & & \{x_3, x_4\} \end{array}$$

The map f updates $\{x_S, x_Q\}$ in each iteration depending on the key bit k_i. Given an $\{x_3, x_4\}$ pair, we need to solve two quadratic equations to compute its preimages. For instance, let $x_4 = x_2^2 + b/x_2^2$, then x_2^2 is the root of $(t^2 + x_4 t + b)$ in $\mathbf{F}$. Since $b \neq 0$, the number of roots are either 0 or 2 with probability 1/2 each. Once we have x_2, we can also solve $x_3 = x_p + (\frac{x_1}{x_1 + x_2})^2 + \frac{x_1}{x_1 + x_2}$ to get x_1. It again has either 0 or 2 roots with equal probability. Thus, for a random $\{x_3, x_4\}$ pair, the expectation of the number of preimages is one.

Denote by $\mathbb{P}^i$ the set of $\{x_S^i, x_Q^i\}$ that lead to a successful check, where x_S^i and x_Q^i denote the value of x_S and x_Q after the i^{th} iteration, respectively. We can divide $\mathbb{P}^i$ into two disjoint subsets: $\mathbb{S}$ and $\mathbb{L}^i$. $\mathbb{S}$ is the set of valid point pairs, $\mathbb{S} = \{\{x(Q), x(Q+P)\}\}$ for all Q on E, while $\mathbb{L}^i$ includes all the rest in $\mathbb{P}^i$. Clearly, all pairs in $\mathbb{S}$ will pass the coherence check. There are in total $\#E$ elements in $\mathbb{S}$.

Consider faults induced during the i^{th} iteration, $0 < i < l-1$. The faults can lead to a successful check if Q_0 and Q_1

are valid points on E, and $Q_1 = Q_0 + P$. In other words, $\mathbb{P}^0 = \mathbb{S}$.

Let $\{x^0(Q), x^0(Q+P)\}$ be an element in $\mathbb{P}^0$. If $Q \notin \{-(2 \cdot P), \mathcal{O}\}$, then $\{x(Q), x(Q+P)\} \in \mathbb{S}$ has either four or no preimages. If $k_i = 0$, then $\{x(Q), x(Q+P)\}$ has four preimages if $Q/2$ exists. Otherwise, $\{x(Q), x(Q+P)\}$ has no preimages. Since $\#E$ has a cofactor 2, only half of $\mathbb{S}$ will have four preimages, and the rest have no preimages. The four preimages of $\{x^0(Q), x^0(Q+P)\}$ are as follows.

$$\begin{aligned} R_A: &\quad \{x_{11}, x_{21}\} \\ R_B: &\quad \{x_{11}, x'_{21}\} \\ R_C: &\quad \{x_{12}, x_{22}\} \\ R_D: &\quad \{x_{12}, x'_{22}\} \end{aligned}$$

Among these four preimages, two are corresponding to two valid pairs: $\{x(Q/2), x(Q/2+P)\}$ and $\{x(Q/2+D), (Q/2+P+D)\}$ where D has order 2. The other two preimages are not corresponding to any valid point pair on E, thus are called *bad* preimages. Thus, $\mathbb{P}^1$ is the union of two sets, $\mathbb{S}$ and $\mathbb{L}^1$, where $\mathbb{L}^1$ contains all the bad preimages. $\mathbb{L}^1$ has the same size of $\mathbb{S}$. In other words, $\mathbb{P}^1$ has $2 \cdot (\#E)$ elements.

We can now calculate the size of $\mathbb{P}^2$. Among the two valid preimages of $\{x^0(Q), x^0(Q+P)\}$, only one of them have preimages, and it again has four images in $\mathbb{P}^2$. The other valid preimage has no preimages. The two bad preimages are randomly distributed in $\mathbf{F} \times \mathbf{F}$, and will have on average two preimages in $\mathbb{P}^2$. Therefore, $\mathbb{P}^2$ includes $\mathbb{P}^1$ and the preimages $\mathbb{L}^1$. The size of $\mathbb{P}^2$ is around $3 \cdot (\#E)$. The same analysis can be applied to compute the size of $\mathbb{P}^i$ for $i > 2$. For example, $\mathbb{P}^{l-2}$ has around $(l-1) \cdot (\#E)$ elements.

In general, the earlier the faults are induced during the scalar multiplication, the larger $\mathbb{P}$ is. Intuitively, faults induced in the last round will fail the check immediately if they don't lead to an valid point pair after one iteration, while faults in the earlier iteration will have more opportunities to hit an valid point pairs.

We can now derive the possibility for a random fault to pass the check. A random fault on $\{x_S, x_Q\}$ in the first iteration yields the highest possibility:

$$Pr \leq \frac{(l-1) \cdot (\#E)}{2^{2m}} \approx \frac{l-1}{2^m}.$$

For a secure ECC implementation, 2^m is relatively large. For example, $m = 163$ is recommended for even constrained devices. In this case, the probability for a random fault to pass the check is negligible.

Power Analysis Note that although the last step of MPL is modified in order to efficiently implement the countermeasure, it preserves its resistance to simple power analysis since the key related operations are still perfectly balanced for both k_0 values (Lines 17-20 in Algorithm 3).

V. Performance Evaluation

This section evaluates the efficiency of the proposed solution, taking into account both performance and area overhead. The performance overhead of the algorithm with the protection is given in Table II.

It shows that there is no overhead in the key related operations except for the last key bit. Since the computations in the last round are performed for both possible values of k_0, the overhead is one point doubling (PD), either $2Q$ or $2S$ (Lines 12-13 of Algorithm 3). Note that addition $Q + S$ is done in both cases and does not need to be computed twice. Since the conversion from the projective to affine coordinates requires a finite field inversion, at least one has to be implemented to convert the final result, even if no protection is implemented. Hence, the inversion of both Q_0 and Q_1 required for the proposed protection scheme comes with the overhead of five field multiplications ($5 \times FM$) (7). Finally, the point addition (PA) for the coherency check $T = Q_1 + P$ and two field multiplications from (8) have to be performed. Thus, the computational overhead of the proposed countermeasure is $1 \times PA + 1 \times PD + 7 \times FM$.

Since the point addition for the coherency check can also be performed using the López-Dahab algorithm, the overhead in the datapath and the control part is negligible. The only area overhead that the countermeasure introduces is storage overhead. It consists of two extra locations for storing X and Z coordinates of the extra point calculated in the Line 10. of Algorithm 3. However, a common Z coordinate system [20], which is often used for restricted devices, reduces the hardware overhead to one extra storing location. Note that this value is temporary and is just used to evaluate the coherency check. It can, depending on actual implementation, be stored in an already existing location which will be overwritten later on.

Comparing to the straight-forward implementation of a coherency check that does not reuse the López-Dahab algorithm to perform the final point addition [5], the proposed implementation is more efficient in both time and area. First, it avoids additional finite field inversions in order to check the coherency at the end of the ECSM. Second, it requires neither recovering the y coordinate of the base point P nor storing it, which make it more efficient than the method proposed in [7]. In addition, since the same point addition is used for both regular and checking part, the proposed method avoids overhead in datapath and the controller part.

VI. Conclusion

In this paper, we presented a new efficient countermeasure to protect ECC implementations utilizing López-Dahab algorithm against fault attacks. The proposed countermeasure detects faults injected during the MPL with a high probability. Instead of adding an additional point addition formula to check the result, we propose to use the already-implemented addition formulas. We observed that the difference between the two points that are used for coherence check can be easily computed. Once this difference is known, López-Dahab can again be applied. The coherence check proposed in this paper uses x-coordinate only. Hence, neither the y coordinates have to be recovered, nor the y coordinate of the base point has to be stored. We believe this method is especially suitable for area-constrained devices.

TABLE II
PERFORMANCE OVERHEAD OF THE PROPOSED COUNTERMEASURE

Referent Design	Operations per key bit		Conv. to Affine	Check
	k_i, $l-1 \geq i \geq 1$	k_0		
Algorithm without Protection	PA+PD	PA+PD	Inversion	-
Algorithm with Protection	PA+PD	PA+2×PD	Inversion+5×FM	PA+2×FM

ACKNOWLEDGMENT

This work was supported in part by K.U.Leuven-BOF (OT/06/40), by the European Commission under contract number ICT-2007-216676 ECRYPT NoE phase II and FP7-238811 UNIQUE Project, by IAP Programme P6/26 BCRYPT of the Belgian State (Belgian Science Policy), by FWO project G.0300.07 and by Austrian Science Fund (FWF) under grant number P22241-N23.

REFERENCES

[1] R. M. Avanzi, H. Cohen, C. Doche, G. Frey, T. Lange, K. Nguyen, and F. Vercauteren, *Handbook of Elliptic and Hyperelliptic Curve Cryptography*. CRC Press, 2005.

[2] I. Biehl, B. Meyer, and V. Müller, "Differential Fault Attacks on Elliptic Curve Cryptosystems," in *CRYPTO*, vol. 1880. Springer, 2000, pp. 131–146.

[3] I. Blake, G. Seroussi, N. Smart, and J. W. S. Cassels, *Advances in Elliptic Curve Cryptography (London Mathematical Society Lecture Note Series)*. New York, NY, USA: Cambridge University Press, 2005.

[4] M. Ciet and M. Joye, "Elliptic Curve Cryptosystems in the Presence of Permanent and Transient Faults," *Des. Codes Cryptography*, vol. 36, no. 1, pp. 33–43, 2005.

[5] A. Dominguez-Oviedo, "On fault-based attacks and countermeasures for elliptic curve cryptosystems," Ph.D. dissertation, University of Waterloo, Canada, 2008.

[6] N. M. Ebeid and R. Lambert, "Securing the Elliptic Curve Montgomery Ladder against Fault Attacks," in *FDTC*, 2009, pp. 46–50.

[7] N. Ebeid and R. Lambert, "Securing the elliptic curve montgomery ladder against fault attacks," in *Proceedings of the 2009 Workshop on Fault Diagnosis and Tolerance in Cryptography*, ser. FDTC '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 46–50. [Online]. Available: http://dx.doi.org/10.1109/FDTC.2009.35

[8] 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 *Hardware-Oriented Security and Trust (HOST), 2010 IEEE International Symposium on*, 2010, pp. 76 –87.

[9] P. Fouque, R. Lercier, D. Réal, and F. Valette, "Fault Attack on Elliptic Curve Montgomery Ladder Implementation," in *Fifth International Workshop on Fault Diagnosis and Tolerance in Cryptography - FDTC'08*. IEEE Computer Society, 2008, pp. 92–98.

[10] P.-A. Fouque, R. Lercier, D. Real, and F. Valette, "Fault attack on elliptic curve montgomery ladder implementation," in *Fault Diagnosis and Tolerance in Cryptography, 2008. FDTC '08. 5th Workshop on*, 2008, pp. 92 –98.

[11] P.-A. Fouque, R. Lercier, D. Réal, and F. Valette, "Fault Attack onElliptic Curve Montgomery Ladder Implementation," in *Fault Diagnosis and Tolerance in Cryptography, Fifth International Workshop, FDTC 2008, Washington DC, USA, August 10, 2008, Proceedings*, L. Breveglieri, S. Gueron, I. Koren, D. Naccache, and J.-P. Seifert, Eds. IEEE Computer Society, August 2008, pp. 92–98.

[12] D. Hein, J. Wolkerstorfer, and N. Felber, "ECC is Ready for RFID - A Proof in Silicon," in *Proceedings of 15th Annual Workshop on Selected Areas in Cryptography (SAC 2008)*, vol. 5381. Lecture Notes in Computer Science, R. Avanzi, L. Keliher and F. Sica (eds.), Springer-Verlag, 2008, pp. 401–413.

[13] D. Johnson, A. Menezes, and S. Vanstone, "The elliptic curve digital signature algorithm (ecdsa)," *International Journal of Information Security*, vol. 1, pp. 36–63, 2001, 10.1007/s102070100002.

[14] M. Joye and S.-M. Yen, "The Montgomery Powering Ladder," in *CHES*, ser. Lecture Notes in Computer Science, vol. 2523. Springer, 2002, pp. 291–302.

[15] O. Kömmerling and M. Kuhn, "Design principles for tamper-resistant smartcard processors," in *USENIX workshop on Smartcard Technology – SmartCard'99*, 1999, pp. 9–20.

[16] J. López and R. Dahab, "Fast multiplication on elliptic curves over gf(2m) without precomputation," in *CHES '99: Proceedings of the First International Workshop on Cryptographic Hardware and Embedded Systems*. London, UK: Springer-Verlag, 1999, pp. 316–327.

[17] D. Naccache, N. P. Smart, and J. Stern, "Projective Coordinates Leak," in *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, ser. Lecture Notes in Computer Science, C. Cachin and J. Camenisch, Eds., vol. 3027. Springer, 2004, pp. 257–267.

[18] C. Research, "Standards for efficient cryptography review of sec1: Elliptic curve cryptography," 2000.

[19] S. M. Yen and M. Joye, "Checking Before Output May Not Be Enough Against Fault-Based Cryptanalysis," *IEEE Trans. Computers*, vol. 49, no. 9, pp. 967–970, 2000.

[20] Yong Ki Lee and Kazuo Sakiyama and Lejla Batina and Ingrid Verbauwhede, "Elliptic-Curve-Based Security Processor for RFID," *IEEE Transactions on Computers*, vol. 57, pp. 1514–1527, 2008.