A comb method to render ECC resistant against Side Channel Attacks

Mustapha Hedabou, Pierre Pinel, Lucien Bénéteau

INSA de Toulouse, LESIA
135, avenue de Rangueil, 31077 Toulouse cedex 4
France

{hedabou, pierre.pinel, lucien.beneteau}@insa-toulouse.fr

Abstract. Side Channel Attacks may exploit leakage information to break cryptosystems on smard card devices. In this paper we present a new SCA-resistant elliptic curve scalar multiplication algorithm, based on the Lim and Lee technique. The proposed algorithm builds a sequence of bit-strings representing the scalar k, characterized by the fact that all bit-strings are different from zero; this property will ensure a uniform computation behaviour for the algorithm, and thus will make it secure against SPA (Simple Power Analysis) attacks. The use of a recently introduced randomization technique achieves the security of the proposed scheme against other SCA attacks. Furthermore, the proposed countermeasures do not penalize the computation time.

Keywords. Elliptic curve cryptosystems, side channel attacks, scalar multiplication, pre-computed table.

1 Introduction

1.1 ECC and side channel attacks

Cryptosystems based on elliptic curves become more and more popular. The security of these cryptosystems is based on the intractability of the discrete logarithm problem on elliptic curves, since no sub-exponential attack is known for a general elliptic curve over a finite field. With a much shorter key length, they offer the same level of security as other public key cryptosystems such as RSA. Thus they seem to be ideal for applications with small resources such a smard cards, mobil devices, etc.

But a new threat has to be tackled with for these systems, as a new class of attacks, called Side Channel Attacks (SCA), has been devised in the last years to allow the adversary to obtain all or part of the secret information embedded in a cryptographic device by observing elements such as computing time or power consumption.

This paper will focus on Simple Power Analysis attacks (SPA), which are based on the analysis of a single execution of the algorithm. This type of attack is particularly efficient on elliptic curve cryptosystems, because the doubling and adding operations on points behave differently and can be easily distinguished by that means. The more sophisticated Differential Power Analysis (DPA) interprets the collected results of several executions of the algorithm with statistical tools. Immunity against DPA attacks may be obtained by combining several data randomization contermeasures.

1.2 Contribution of this paper

In this paper we propose a new SCA-resistant scheme based on the Lim and Lee [LL94] technique. The proposed scheme constructs a new sequence of bit-strings representing the scalar k from its bit-string sequence as introduced by Lim and Lee. This new sequence is characterized by the fact that all its bit-strings are different from zero. The proposed scheme is more efficient than Möller's one [Möl01], its cost being about 5% to 10% smaller than Möller's one.

This paper is organized as follows: section 2 briefly reviews the mathematical background for elliptic curve cryptography. In section 3, we introduce the Side Channel Attacks and the countermeasures against them. In section 4, we describe Lim and Lee's algorithm and its shortfalls, before introducing our method and studying its efficiency and resistance against side channel attacks. Finally, we conclude in section 5.

2 Mathematical background

2.1 Elliptic curves

An elliptic curve is the set of the solutions of a Weierstrass equation over a field. For cryptographic purposes, this field is most frequently a finite field of the form $GF(p)$ or $GF(2^m)$. In these particular cases, the Weierstrass equation can be reduced to the following simpler forms

$y^2 = x^3 + ax + b$ over $GF(p)$, with $a, b \in GF(p)$ and $4a^3 + 27b^2 \neq 0$,

$y^2 + xy = x^3 + ax^2 + b$ over $GF(2^m)$, with $a, b \in GF(2^m)$ and $b \neq 0$.

If the formal point at infinity Θ is added to the set of solutions, an adding operation can be defined over the elliptic curve, which turns the set of the points of the curve into a group.

The adding operation between two points is defined as follows.

Let $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$ be two points on the elliptic curve, neither being the point at infinity.
Over $GF(p)$ the inverse of a point P_1 is $-P_1 = (x_1, -y_1)$. If $P_1 \neq -P_2$ then $P_1 + P_2 = P_3 = (x_3, y_3)$ with

$$x_3 = \lambda^2 - x_1 - x_2,$$

$$y_3 = \lambda(x_1 - x_3) - y_1$$

and

$$\lambda = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1} \text{ if } P_1 \neq P_2 & \text{(adding)} \\ \frac{3{x_1}^2 + a}{2y_1} \text{ if } P_1 = P_2 & \text{(doubling)} \end{cases}$$

Over $GF(2^m)$ the inverse of a point P_1 is $-P_1 = (x_1, x_1 + y_1)$. If $P_1 \neq -P_2$ then $P_1 + P_2 = P_3 = (x_3, y_3)$ with

$$x_3 = \lambda^2 + \lambda + x_1 + x_2 + a,$$

$$y_3 = \lambda(x_1 + x_3) + x_3 + y_1,$$

and

$$\lambda = \begin{cases} \frac{y_2 + y_1}{x_2 + x_1} \text{ if } P_1 \neq P_2, & \text{(adding)} \\ x_1 + \frac{y_1}{x_1} \text{ if } P_1 = P_2. & \text{(doubling)} \end{cases}$$

2.2 Projective coordinate representations

All preceeding formulas for adding and doubling of elliptic curve point require a field inversion. Unfortunately, this operation in known to be highly expensive. To avoid the corresponding expenses, it may be advantageous to represent the points using projective coordinates.

In standard projective coordinates, the projective point $(X,\ Y,\ Z)$, $Z \neq 0$, corresponds to the affine point $(\frac{X}{Z}, \frac{Y}{Z})$. The projective equations of the curve are then

$$Y^2 Z = X^3 + aXZ^2 + bZ^3 \text{ over } GF(p),$$

$$Y^2 Z + XYZ = X^3 + aX^2 Z + bZ^3 \text{ over } GF(2^m).$$

In Jacobian projective coordinates, the projective point $(X,\ Y,\ Z)$, $Z \neq 0$, corresponds to the affine point $(\frac{X}{Z^2}, \frac{Y}{Z^3})$. The projective equations are:

$$Y^2 = X^3 + aXZ^4 + bZ^6 \text{ over } GF(p),$$

$$Y^2 Z + XYZ = X^3 + aX^2 Z^2 + bZ^6 \text{ over } GF(2^m).$$

The randomized Jacobian coordinates method transforms the point $(x, y, 1)$ in projective coordinates into $(r^2 x, r^3 y, r)$ using a random non-zero number. The properties of the Jacobian coordinates imply that $(x, y, 1) = (r^2 x, r^3 y, r)$.

3 Side Channel Attacks

The basic algorithm for computing $Q = kP$ on an elliptic curve is based on doubling and adding operations.

Algorithm 1 : Simple doubling-and-adding algorithm
Input: $P,\ k = (k_{l-1}, ..., k_0)_2$.
Output: $Q = kP$.
1. $Q = P$.
2. For $i = l - 2$ *downto 0 do*
2.1 If $k_i = 0$ *then* $Q \leftarrow 2Q$
2.2 else $Q \leftarrow 2Q + P$.
3. Return Q.

The two cases which may occur in step 2 of this algorithm can be distinguished by a simple analysis of power consumption (SPA attack): when the power trace shows only doubling, we conclude that $k_i = 0$, and that $k_i = 1$ in the other case.

Improved point multiplication algorithms such as the m-ary or the sliding window methods will obscure k to some degree, but plenty of information may still be revealed.

To prevent side channel attacks, we can incorporate in the implementation some countermeasures intended to make the processing time of the algorithm independent from the data. These standard countermeasures consist usually in

· Performing some dummy operations [Cor99].
· Using data randomization [LS01, JQ01, JT01, Cor99]
· Using specific algorithms [LD99, OS02, Möl01, Mon87]

In [Cor99], the author suggests to add dummy operations and to use data randomization to obtain a resistant algorithm against SCA attacks. The inconvenient of this method is that it penalizes the running time.

Liadet and Smart [LS01] have proposed to reduce information leakage by using a special point representation in some elliptic curves pertaining to a particular category, such that a single formula can be used for adding and doubling operations. In [JQ01], Joye and Quisquater suggest to use the Hessian form, but their parametrization is not fully general. A common disadvantage of this type of methods is that they require to use some specific elliptic curves. All curves suitable for [LS01] have a group order divisible by 4, and curves suitable for [JQ01] have a group order divisible by 3. None of these methods is applicable to the curves recommended by the NIST and the SECG as given in [NIST00, SEC00].

The third approach to prevent SPA attacks is to develop special types of algorithms, like Montgomery's [Mon87] and Möller's [Möl01] algorithms. In this paper, we will present a new SCA-resistant algorithm, which works on general elliptic curves.

4 The proposed SCA-resistant algorithm

4.1 The fixed-base comb method and its shortcomings

In the following we describe the fixed-base comb method [BHLM01] based on the Lim and Lee [LL94] technique.
Let $(k_{l-1}, \cdots, k_1, k_0)$ be the binary representation of an integer k, ie $k = \sum_{i=0}^{i=l-1} k_i 2^i$, with $k_i \in \{0, 1\}$, and let w be an integer such as $w \geq 2$; we set $d = \lceil \frac{l}{w} \rceil$.
P being an elliptic curve point, for all $(b_{w-1}, \cdots, b_1, b_0) \in \mathbb{Z}_2{}^w$, we define $[b_0, b_1, \cdots, b_{w-1}]P = b_0 + b_1 2^d + b_2 2^{2d} + \cdots + b_{w-1} 2^{(w-1)d}$.

The comb method considers that k is represented by a matrix of w rows and d columns, and processes k columnwise.

Algorithm 2 : Fixed-base comb method
Input: a positive integer $k = (k_{l-1}, \cdots, k_1, k_0)$*, an elliptic curve point* P *and a window width* w *such as* $w \geq 2$.
Output: kP.
1. $d = \lceil \frac{l}{w} \rceil$.
2. Precomputation: compute $[b_{w-1}, \cdots, b_1, b_0]P$ *for all* $(b_{w-1}, \cdots, b_1, b_0) \in \mathbb{Z}_2{}^w$.
3. By padding k *on the left with 0's if necessary, write* $k = K^{w-1} \parallel \cdots \parallel K^1 \parallel K^0$*, where each* K^j *is a bit-string of length* d*. Let* K_i^j *denote the* i*-th bit of* K^j.
4. $Q \leftarrow [K_{l-1}^{w-1}, \cdots, K_{l-1}^1, K_{l-1}^0]P$.
5. for i *from* $d-2$ *down to* 0 *do*
  5.1 $Q \leftarrow 2Q$
  5.2 $Q \leftarrow Q + [K_i^{w-1}, \cdots, K_i^1, K_i^0]P$.
6. Return Q.

The execution of a SPA attack on the fixed-base comb method can allow to detect some information on the bits of the secret scalar. Indeed, the comb method performs an adding and doubling operation if the bit-string $[K_i^{w-1}, \cdots, K_i^1, K_i^0]$ is different from 0, and only a doubling operation in the other case; thus the analysis of the power consumption's measures during the execution of the algorithm can reveal whether the bit-string $[K_i^{w-1}, \cdots, K_i^1, K_i^0]$ is zero (ie $(K_i^{w-1}, \cdots, K_i^1, K_i^0) = (0, \cdots, 0)$) or not. Since the probability to have zero bit-string ($[K_i^{w-1}, \cdots, K_i^1, K_i^0] = 0$) is less important than the probability to have a single zero bit ($k_i = 0$), the comb method offers a better resistance than the binary method, but it is not totally secure against SPA attacks.

In the next section we will convert the comb method to an SPA-resistant scheme by constructing a new sequence of non-zero bit-strings. The obtained scheme combined with randomization countermeasures achieves the security against SCA attacks.

4.2 Proposed method

We propose to modify the bit-string representation of k so as to make the fixed-base comb method secure against SPA. We want to generate a sequence of bit-strings representing k so that every bit-string is different from zero.

Each new bit-string, noted $(\mathbb{K}^i, s_i)$, will result from the composition of some bit-string $[K_j^{w-1}, \cdots, K_j^1, K_j^0]$, with $j \leq i$, and of one more bit noted $s_i = \pm 1$, which will be equal to 1 if $\mathbb{K}^i = [K_j^{w-1}, \cdots, K_j^1, K_j^0]$, and equal to -1 if $\mathbb{K}^i = -[K_j^{w-1}, \cdots, K_j^1, K_j^0]$.

As we will see later, we will only need to convert odd integers into sequences of bit-strings. We shall hence assume in the next subsection that k is odd.

4.2.1 Computing of the new bit-string sequence for odd k's

A first idea would be to do the conversion by initializing $s_0 = 1$ and constructing the new sequence of bit-strings $(\mathbb{K}_i, s_1)$ from the bit-strings $[K_i^{w-1}, \cdots, K_i^1, K_i^0]$ by setting

$$\begin{cases} (\mathbb{K}_i, s_i) = ([K_{i-1}^{w-1}, \cdots, K_{i-1}^1, K_{i-1}^0], s_{i-1}) \\ (\mathbb{K}_{i-1}, s_{i-1}) = ([K_{i-1}^{w-1}, \cdots, K_{i-1}^1, K_{i-1}^0], -s_{i-1}) \end{cases}$$

if $[K_i^{w-1}, \cdots, K_i^1, K_i^0] = 0$ and

$$\begin{cases} (\mathbb{K}_i, s_i) = ([K_i^{w-1}, \cdots, K_i^1, K_i^0], s_{i-1}) \\ (\mathbb{K}_{i-1}, s_{i-1}) = ([K_{i-1}^{w-1}, \cdots, K_{i-1}^1, K_{i-1}^0], s_{i-1}) \end{cases}$$

otherwise.

But in this version of the algorithm constructing the sequence of non-zero bit-strings representing k, a bit-string of k would either be touched if it is a zero bit-string or kept unchanged otherwise; hence a SPA attack against the algorithm is conceivable. To deal with this threat, we present now a SPA-resistant algorithm for recording the new sequence of bit-strings of the scalar k.

Algorithm 3 : Computing of the new bit-strings representing k
Inputs: an odd positive integer $k = (k_0, k_1, , \cdots, k_{l-1})$, and a window width $w \geq 2$.
Output: the sequence of bit-strings $((\mathbb{K}_o, s_0), (\mathbb{K}_1, s_1), \cdots, (\mathbb{K}_{d-1}, s_{d-1}))$.
1. $d = \lceil \frac{l}{w} \rceil$.
2. For $i = l$ to $(wd - 1)$ do $k_i \leftarrow 0$.
3. $(\mathbb{K}_0, s_0) \leftarrow ([K_0^{w-1}, \cdots, K_0^1, K_0^0], 1)$.
5. For $i = 1$ to $i = d - 1$ do
5.1 If $[K_i^{w-1}, \cdots, K_i^1, K_i^0] \neq 0$ then $c_i \leftarrow 1$ else $c_i \leftarrow 0$.
5.2 $b[0] \leftarrow [K_{i-1}^{w-1}, \cdots, K_{i-1}^1, K_{i-1}^0]$, $b[1] \leftarrow [K_i^{w-1}, \cdots, K_i^1, K_i^0]$.
5.3 $s[0] \leftarrow -1$, $s[1] \leftarrow 1$.
5.4 $(\mathbb{K}_i, s_i) \leftarrow (b[c_i], 1)$, $(\mathbb{K}_{i-1}, s_{i-1}) \leftarrow (b[0], s[c_i])$.
6 Return $(\mathbb{K}_o, s_0), (\mathbb{K}_1, s_1), \cdots, (\mathbb{K}_{d-1}, s_{d-1})$.

We prove hereunder that our new sequence of bit-strings represents correctly k.

Theorem 1 *Algorithm 3, when given an odd scalar k, outputs a sequence of bit-strings $(\mathbb{K}_i, s_i)$ such as $\sum_{i=0}^{d-1} 2^i s_i \mathbb{K}_i = \sum_{i=0}^{i=d-1} 2^j [K_i^{w-1}, \cdots, K_i^1, K_i^0] = k$.*

Proof. We assume that the equality
$Q_i = \sum_{j=0}^{j=i} 2^j s_j \mathbb{K}_j = \sum_{j=0}^{j=i} 2^j [K_j^{w-1}, \cdots, K_j^1, K_j^0]$ holds for each $i \in \{0, \cdots, n-1\}$. We have to check that the equality is also satisfied for n, i.e that $Q_n = \sum_{j=0}^{j=n} 2^j s_j \mathbb{K}_j = \sum_{j=0}^{j=n} 2^j [K_j^{w-1}, \cdots, K_j^1, K_j^0]$.

First, if $[K_n^{w-1}, \cdots, K_n^1, K_n^0]$ is different from zero, we have $\mathbb{K}_n = [K_n^{w-1}, \cdots, K_n^1, K_n^0]$ and $s_n = 1$. Given that $Q_n = Q_{n-1} + 2^n s_n \mathbb{K}_n$ and applying the induction hypothesis to Q_{n-1}, the equality is verified.

Now, if $[K_n^{w-1}, \cdots, K_n^1, K_n^0]$ is equal to zero, we have $\mathbb{K}_n = [K_{n-1}^{w-1}, \cdots, K_{n-1}^1, K_{n-1}^0]$, $s_n = 1$ and $\mathbb{K}_{n-1} = [K_{n-1}^{w-1}, \cdots, K_{n-1}^1, K_{n-1}^0]$, $s_{n-1} = -1$. Thus

$$Q_n = \sum_{j=0}^{j=n} 2^j s_j \mathbb{K}_j = \sum_{j=0}^{j=n-2} 2^j s_j \mathbb{K}_j + 2^{n-1} s_{n-1} \mathbb{K}_{n-1} + 2^n s_n \mathbb{K}_n$$
$$= \sum_{j=0}^{j=n-2} 2^j s_j \mathbb{K}_j - 2^{n-1} [K_{n-1}^{w-1}, \cdots, K_{n-1}^1, K_{n-1}^0] + 2^n [K_{n-1}^{w-1}, \cdots, K_{n-1}^1, K_{n-1}^0].$$

By applying the induction hypothesis to Q_{n-2}, we have
$Q_{n-2} = \sum_{j=0}^{j=n-2} 2^j s_j \mathbb{K}_j = \sum_{j=0}^{j=n-2} 2^j [K_j^{w-1}, \cdots, K_j^1, K_j^0]$; thus

$$Q_n = \sum_{j=0}^{j=n} 2^j s_j \mathbb{K}_j = \sum_{j=0}^{j=n-2} 2^j [K_j^{w-1}, \cdots, K_j^1, K_j^0] + 2^{n-1} [K_{n-1}^{w-1}, \cdots, K_{n-1}^1, K_{n-1}^0]$$
$$= \sum_{j=0}^{j=n} 2^j [K_j^{w-1}, \cdots, K_j^1, K_j^0].$$

The equality is satisfied, which completes the proof.

4.2.2 Computation of kP

The following algorithm, based on the new bit-string representation of k, implements a scalar multiplication secure against simple power analysis on a general elliptic curve.

As Algorithm 3 supposes that k is odd, we propose to replace the computation of kP by the computation of $((k + 1)P - P)$ if k is even. But this implies that one more point adding operation is performed if k is even, which would give way to a SPA attack detecting whether the secret scalar is even or not. To deal with this threat, we replace also the computation of kP if k is odd, calculating $((k + 2)P - 2P)$ instead of kP in this case.

In case of an even scalar, one more adding operation is performed, an SPA attack can then detect whether the secret scalar is even or not. To deal with this threat, we propose to convert as well the odd scalar k to $k' = k + 2$. Thus the scalar multiplication kP is recovered by performing the substraction of either $k'P - P$ if k is even, and $k'P - 2P$ otherwise.

Algorithm 4 : Computation of kP using the new bit-string representation of k

Input: a scalar k and an elliptic curve point P.
Output: $Q = kP$.
1. (Precomputation) Compute $[b_{w-1}, \cdots, b_1, b_0]P$ for all $(b_{w-1}, \cdots, b_1, b_0) \in \mathbb{Z}_2{}^w$.
2. If k mod $2 = 0$ then $k' \leftarrow k + 1$ else $k' \leftarrow k + 2$.
3. Compute the sequence of bit-strings $(\mathbb{K}'_0, s_0), (\mathbb{K}'_1, s_1), \cdots, (\mathbb{K}'_{d-1}, s_{d-1})$ representing k'.
4. $Q \leftarrow \mathbb{K}'_{d-1}P$. // (each $\mathbb{K}'_j P$ will be fetched from the precomputed table)
5. For i from $d-2$ down to 0 do
  5.1 $Q_1 \leftarrow 2Q$
  5.2 $Q \leftarrow Q_1 + s_i \mathbb{K}'_i P$
6. $P' \leftarrow 2P$.
7. If k mod $2 = 0$ then return $Q - P$ else return $Q - P'$.

4.3 Security against SPA attacks

As shown before, all bit-strings representing k are different from zero. Thus, the proposed algorithm computes the scalar multiplication with a uniform behaviour, by performing exactly an adding and doubling operation at each step. Consequently, the execution of a SPA attack against the proposed algorithm can not reveal any information on the bits of the secret scalar k.

4.4 Randomization countermeasures against other CSA attacks

The use of projective randomization techniques such as randomized projective coordinates [Cor99] or random isomorphic curves [JT01] is known to prevent DPA attacks.

But Okeya and Sakurai's second-order DPA attack [OS02], which they have proposed against Möller's window method [Möl01], might still be applied against our algorithm. This attack exploits the correlation between the power consumption and the weight of the loaded data. A precomputed table is used by our algorithm, which is accessed for each digit i to get some point $\mathbb{K}'_i P$ to be added to Q_1. We have to take care of the fact that an attacker, by monitoring the power consumption, could manage to detect whether or not $\mathbb{K}'_i$ is equal to $\mathbb{K}'_j$. To avoid this possiblity, we change the randomization of each precomputed point after getting the point in the table. Thus, even if we have got the same point for different digits, the new point randomization implies that we load a different data.

Exploiting the fact that the points with 0-coordinates $(x, 0)$ or *or* $(0, y)$) can not be randomized by the usual randomisation techniques, Geiselmann and Steinwandt [GS04] have proposed another efficient attack on Möller's method, which is also efficient on the fixed-base comb method. But recently Itoh and al [IIT04] have presented the Randomized Linearly-transformed Coordinates, which allow to randomize all intermediate values. This countermeasure prevents the Geiselmann-Steinwandt attack, and also the more powerful RPA and ZPA attacks [Gou03, AT03].

4.5 Efficiency

The main phase of the proposed algorithm computes an adding and doubling operation at each step. The cost of this phase is $(d-1)(A+D)$, where A, and D denote the adding and doubling point operation. Since we have to perform $Q-P$ or $Q-2P$ to recover the multiplication result kP, the total cost is $d(A+D)$.

Now, we evaluate the cost of precomputation phase. In this phase, we generate the sequence of points $[b_{w-1}, \cdots, b_1, b_0]P$, for all $(b_{w-1}, \cdots, b_1, b_0) \in \mathbf{Z}_2{}^w$, such as

$$[b_{w-1}, \cdots, b_1, b_0]P = b_{w-1}2^{(w-1)d}P + \cdots + b_2 2^{2d}P + b_1 2^d + b_0 P.$$

To perform the precomputing phase, we first compute $2^d P, 2^{2d} P, \cdots, 2^{(w-1)d}P$, which will cost $(w-1)d$ doubling operations. The second step consists in computing all possible combinations $\sum_{i=r}^{i=s} b_1 2^{id}P$, with $b_i \in \{0,1\}$, and $0 \leq r < s \leq w-1$. The number of these combinations is $2^w - w$. Thus, the cost of this second step is $2^w - w$ adding operations, and the total cost of the proposed method, including the efforts to prevent SPA attacks, is

$$wdD + (2^w - w + d)A.$$

To prevent second-order DPA attacks, we have to rerandomize each precomputed point; this will cost $5dM$, where M denotes a field multiplication.

The proposed method perfoms only one more point adding and doubling than the fixed-base comb method, which is negligible.

5 Conclusion

In this paper, we have presented a new SCA-resistant method for computing elliptic curve scalar multiplications based on the fixed-base comb method. The proposed method first converts the comb method to an SPA-resistant scheme and then combines it with Randomized Linearly-transformed Coordinates to achieve the security against other SCA attacks. Furthermore, this is done without increasing significantly the amount of computation time, as the proposed method performs only one more point adding and doubling than the fixed-base comb method.

References

[AT03] T.Akishita, T.Takagi. *Zero-value point attacks on elliptic curve crytosystems.* In: ISC 2003, vol 2851, Lecture Notes in Computer Science (LNCS), pp. 218-233, Springer-Verlag, 2003.

[BHLM01] M.Brown, D.Hankerson, J.Lopez, A.Menezes. *Software implementation of the NIST elliptic curves over prime fields.* In: Progress in Cryptology CT-RSA 2001, D.Naccache, editor, vol 2020, LNCS, pp. 250-265, 2001.

[CMO98] H.Cohen, A.Miyaji, T.Ono. *Efficient elliptic curve exponentiation using mixed coordinates.* In: Advances in cryptology-Asiacrypt98, K.Ohta and D.Pei, editors, vol 1514, LNCS, pp. 51-65, 1998.

[Cor99] J.S.Coron. *Resistance against differential power analysis for elliptic curve cryptosystems.* In: Cryptography Hardware and Embedded Systems-CHES'99, C.K. Koç and C.Paar, editors, vol 1717, LNCS, pp. 292-302, 1999.

[GS04] W.Geiselmann, R.Steinwandt. *Power attacks on a side-channel resistant elliptic curve implementation,* Information Processing Letters, volume 91, pp. 29-32, 2004.

[Gou03] L.Goubin. *A refined power-analysis attacks on elliptic curve cryptosystems.* In: PCK 2003, vol 2567, LNCS, pp. 199-210, 2003.

[IIT04] K.Itoh, T.Izu, M.Takenaka. *Efficient countermeasures against power analysis for elliptic curve cryptosystems.* In: Proceedings of CARDIS-WCC 2004.

[JQ01] M. Joye, J.J. Quisquater. *Hessian elliptic curves and side-channel attacks.* In: Cryptography Hardware and Embedded Systems-CHES'01, C.K. Koç, D. Naccache and C.Paar, editors, vol 2162, LNCS, pp. 412-420, 2001.

[JT01] M. Joye, C. Tymen. *Protections against differential analysis for elliptic curve cryptography: an algebraic approach.* In: Cryptography Hardware and Embedded Systems-CHES'01, C. Koç and D. Naccache and C. Paar, editors, vol 2162, LNCS, pp. 386-400, 2001.

[KJJ99] P. Kocher, J. Jaffe, B. Jun. *Differential power analysis.* In: Advances in Cryptology-CRYPTO'99, M. Wiener, editor, vol 1666, LNCS, pp. 388-397, 1999.

[Koc96] P. Kocher. *Timing attacks on implementation of Diffie-Hellman, RSA, DSA and other systems.* In: Advances in Cryptology-CRYPTO'96, N.Koblitz, editor, vol 1109, LNCS, pp. 104-113, 1996.

[LD99] J. Lopez, R. Dahab. *Fast multiplication on elliptic curves over $GF(2^m)$ without precomputation.* In: Cryptography Hardware and Embedded Systems CHES'99, C. Koç and C. Paar, editors, vol 1717, LNCS, pp. 316-327, 1999.

[LL94] C.Lim and P.Lee. *More flexible exponentiation with precomputation.* In: Advances in Cryptology-CRYPTO'94, vol 839, LNCS, pp. 95-107, 1994.

[LS01] P.V. Liardet, N. Smart. *Preventing SPA/DPA in ECC systems using the Jacobi form.* In Cryptography Hardware and Embedded Systems-CHES'01, C. Koç, D. Naccache and C. Paar, editors, vol 2162, LNCS, pp. 401-411, 2001.

[Möl01] B. Möller. *Securing elliptic curve point multiplication against side-channel attacks.* In: Information Security, G.I Davida and Y. Frankel, editors, vol 2200, LNCS, pp. 324-334, 2001.

[Mon87] P.L. Montgomery. *Speeding up the Pollard and elliptic curve methods of factorization,* Mathematics of Computation, 48(177), pp. 243-264, January 1987.

[NIST00] National Institute of Standard and Technology (NIST). *Digital signature standard (DSS).* FIPS PUB 186-2, 2000

[OS02] K. Okeya, K. Sakurai. *A Second-Order DPA attacks breaks a window-method based countermeasure against side channel attacks,* Information Security Conference (ISC 2002), LNCS 2433, pp. 389-401, 2002.

[SEC00] Certicom research. *Standard for efficient cryptography.* Version 1.0, 2000. Available at url http:/www.secg.org/.