A Fast Parallel Scalar Multiplication against Side–Channel Analysis for Elliptic Curve Cryptosystem over Prime Fields*

Dabi Zou[1,2] and Dongdai Lin[1]

[1] Department of Computer Science and Technology,
University of Science and Technology of China
[2] State Key Laboratory of Information Security,
Software Institute of China Academic of Sciences
{dbzou, ddlin}@is.iscas.ac.cn

Abstract

The scalar multiplication is the dominant operation in Elliptic Curve Cryptosystems (ECC). In this paper, we propose a modified width–w window method to compute the scalar multiplication efficiently and securely against side–channel analysis, based on the side–channel atomicity introduced by Benoit Chevallier–Mames. Utilizing this window method, we propose a new parallel scalar multiplication algorithm, which is secure against side–channel analysis and more efficient than existing ones.

Keywords: Elliptic curve cryptosystem; side-channel analysis; window method; parallel scalar multiplication

1 Introduction

ECC introduced by Victor Miller [2] and Neal Koblitz [3] independently, have recently gained a lot of attention in industry and academia. The main reason for this is the fact that there is no sub–exponential algorithm known to solve the discrete logarithm problem on a properly chosen elliptic curve. This means that significantly smaller parameters can be used in ECC than in other competitive systems such as RSA, while with equivalent levels of

*Supported by the National Natural Science Foundation of China under Grant No.90204016; the National High-Tech Research and Development Plan of China under Grant No.2003AA144030; the National Grand Fundamental Research 973 Program of China under Grant No.2004CB318004.

security. A typical example of the size in bits of the keys used in different public–key systems, with a comparable level of security (against known attacks), is that a 160–bit ECC key is equivalent to RSA with a modulus of 1024 bits. This makes ECC ideal for constrained environments such as PDAs and smart cards.

The fundamental operation in ECC is scalar multiplication, namely, given an integer d and an elliptic curve point P, the computation of dP. It is computed by a series of doubling(ECDBL) and addition(ECADD) operations of the point P, depending upon the bit sequence representing d. A lot of methods have been proposed to perform the scalar multiplication in a secure and efficient way. For an excellent review, see [16]. In this paper, we propose a secure and more efficient method (the SPA–resistant width–w window method.) for the computing of scalar multiplication, which is based on the side–channel atomicity [15] and the width–w window method. After that, we propose a parallel scalar multiplication algorithm integrated with the SPA–resistant width–w window method, which is more efficient than existing ones and is SPA–resistant.

The rest of this paper is organized as follows. In the next section, we briefly introduce the necessary backgrounds, which include elliptic curve preliminaries, non–adjacent form (NAF) and width–w window method using NAF, side–channel analysis, and side–channel atomicity. The proposed SPA–resistant width–w window method is described in section 3 and the parallel scalar multiplication algorithm is introduced in section 4. Finally in section 5, we give a brief conclusion.

2 Background

2.1 Elliptic curve preliminaries

There exists an extensive literature on elliptic curve cryptography. Here we just mention the results we needed, without proof. For detailed descriptions, we refer the reader to [16].

In the current work we will concentrate on curves over large prime fields only.

Consider the elliptic curve E defined over a prime field F_p (with $p > 3$) given by the following equation:

$$E/_{F_p} : y^2 = x^3 + ax + b \tag{1}$$

To avoid field inversion, which is time consuming, Jacobian coordinates are generally used for representing point on $E/_{F_p}$. With Jacobian coordinates, the doubling of P is $2(X_1, Y_1, Z_1) = (X_3, Y_3, Z_3)$,where

$$X_3 = M^2 - 2S, \quad Y_3 = M(s - X_3) - T, \quad Z_3 = 2Y_1Z_1 \tag{2}$$

With $M = 3X_1^2 + aZ_1^4, S = 4X_1Y_1^2$ and $T = 8Y_1^4$.

The sum of two (distinct) points $P = (X_1, Y_1, Z_1)$ and $Q = (X_2, Y_2, Z_2)$ is (X_3, Y_3, Z_3), where

$$\begin{aligned} X_3 &= W^3 - 2U_1W^2 + R^2, \\ Y3 &= -S_1W^3 + R(U_1W^2 - X_3), \\ Z_3 &= Z_1Z_2W \end{aligned} \tag{3}$$

With $U_1 = X_1Z_2^2, U_2 = X_2Z_1^2, S_1 = Y_1Z_2^3, S_2 = Y_2Z_1^3, W = U_1 - U_2$ and $R = S_1 - S_2$.

Let $[a], [m]$ and $[s]$ denote the time required for one addition, multiplication and squaring in the underlying field respectively. Then ADD has complexity of $7[a] + 12[m] + 4[s]$ and DBL has $11[a] + 4[m] + 6[s]$([17]).

The scalar multiplication is generally computed using a left-to-right binary algorithm(see Algorithm 1).

Algorithm 1: Binary algorithm for scalar multiplication

```
Input: integer $d = d_{n-1}2^{n-1} + \cdots + d_1 2 + d_0, d_{n-1} \neq 0$,
       and a base point $P$
Output: $dP$
1: $P_0 := p$ ;
2: for $i := 0$ to $n - 2$
3: $P_{i+1} = \text{ECDBL}(P_i)$;
4: if $d_{n-2-i} = 1$
5:    $P_{i+1} = \text{ECADD}(P_{i+1}, P)$;
6: Return($P_{n-1}$);
```

2.2 Non-adjacent form(NAF)

Every integer d has a unique representation of the form $d = \sum_{j=0}^{l-1} d_j 2^j$, where each $k_j \in \{-1, 0, 1\}$,such that no two consecutive digits are nonzero. This representation is the so called non-adjacent form (NAF), which was first described by Reitwiesner [1] (see also [7] and [11]).

The computing of dP can be improved if d is represented in NAF. This is obtained from the following facts:

- The expected weight of a NAF of length l is $l/3$, see[7] for details.
- The computation of the negation of a point $P(x, y)$ can be obtained easily: $-P = (x, -y)$, so the cost of addition or subtraction is practically the same.

There are several algorithms for computing the NAF of d from its binary representation. Algorithm 2 is from Solinas [9].

Algorithm 2: Computation of $NAF(d)$

```
Input: integer $d$
Output: $NAF(d) = (d_{l-1} \cdots d_1 d_0)$
1:  c:=d;   l:=0;
2:  while $c > 0$ do
3:     if c odd then $\{d_l := 2 - (c \bmod 4);$   $c := c - d_l;\}$
4:     else $d_l := 0;$
5:     c:=c/2; l:=l+1;
6:  return (NAF($d$) = $(d_{l-1} \cdots d_1 d_0)$);
```

2.3 The width-w window method using NAF

There are several generalizations of the binary method such as the m–ary method, sliding method, etc., work by processing simultaneously a block of digits. Here we describe the so called width–w window method(see [9]). In order to utilize the non–adjacent form in the width–w window method, it is necessary to compute the width–w non–adjacent form $d = \sum_{j=0}^{l-1} d_j 2^j$ where:

- each nonzero d_j is odd and less than 2^{w-1} in absolute value, w is an integer greater than 1
- among any w consecutive coefficients, at most one is nonzero.

The algorithm for computing the width–w non–adjacent form of d is given below (Algorithm 3, see [9]). So the scalar multiplication can be carried out by Algorithm 4.

Algorithm 3: Computation of $NAF_w(d)$

```
Input: integer $d$
Output: $NAF_w(d) = (d_{l-1} \cdots d_1 d_0)$
1:  $c := d;$   $l := 0;$
2:  while $c > 0$do
3:     if c odd then
4:        $d_l := 2 - c \bmod 2^w;$
5:        if $d_l > 2^{w-1}$ then $d_l := d_l - 2^w$
6:        $c := c - d_l;$
7:     else $d_l := 0;$
8:     $c := c/2;$   $l := l + 1;$
9:  return $(NAF_w(d) := (d_{l-1} \cdots d_1 d_0))$
```

Algorithm 4: The width-w window method

```
Input: integers $d$ and window width-$w$, and a point $P = (x, y) \in \mathrm{E}(F_q)$
Output: the point $Q = dP \in \mathrm{E}(F_q)$
//Precomputation: compute $uP$ for $u$ odd and $2 < u < 2^{w-1}$
1:  $P_0 := P;$   $T := 2P;$
```

```
2: for i from 1 to $2_{w-2} - 1$ do { $P_i := P_{i-1} + T$; }
***//Main computation***
3: compute $NAF_w(d) := (d_{l-1} \cdots d_1 d_0)$;   Q:=(0,0);
4: for j from $l-1$ downto 0 do
5:    Q:=2Q;
6:    if $d_j \neq 0$ then $i = (|d_j| - 1)/2$;
7:       if $d_j > 0$then $Q := Q + P_i$;
8:       else $Q := Q - P_i$;
9: return Q;
```

The number of nonzero digits in the $NAF_w(d)$ is on average $l/(w+1)$ [10]. Therefore, Algorithm 4 requires $2^{w-2} - 1$ additions and one doubling for the precomputation step, and $l/(w+1)$ additions and $l-1$ doublings for the main computation. So the total time complexity of Algorithm 4 is

$$(2^{w-2} - 1 + l/(w+1))[ECADD] + l[ECDBL]$$

2.4 Side-Channel Analysis and Side Channel Atomicity

Side-Channel Analysis

Side channel analysis(SCA) was introduced for the first time by Kocher in 1996[5], and was further studied in 1999[6]. This attack allow adversaries to obtain partial information on a cryptographic device, or even the secret key in it, by observing side channel information such as computing time and power consumption traces if the implementation is poor. This is a serious and practical threat especially to mobile devices like smart cards. Thus, it is necessary to implement the algorithms not only efficient but also SCA-resistant.

In particular, simple power analysis(SPA) and differential power analysis(DPA) are two kinds of such attack. SPA utilizes information from a single computation, while DPA uses statistical tools to evaluate information from multiple computations. There are several ways to resist SPA. Most of these methods are on the expense of efficiency, like Montgomery ladder, while [15] propose the least amount of computational overhead very recently(see next paragraph). In order to thwart the DPA, one has to randomize the inputs of the crypto–algorithm so that it is no longer readily for the attacker to prepare two sets of points with a selection function. And the randomization includes base–point P's and scalar d's randomizing. For more details, see [14].

Side Channel Atomicity

The main idea of Side–channel atomicity [15] is to divide each EC–operation, namely ECADD and ECDBL, into atomic blocks which are indistinguishable from the side–channel. So, in this model the computation of scalar multiplication is a sequence of indistinguishable atomic blocks. By this way, the attacker cannot use simple power analysis to obtain the side channel information, and thus this implementation is SPA–resistant. The following algorithm (Algorithm 5) is such an implementation [15].

Algorithm 5: Side-channel atomic double–and–add algorithm for elliptic curves over F_p

```
    Input: $P := (X, Y, Z)$,$d$,matrix($u^*_{k,l}$)
    Output: $P_d = dP$
1: $R_1 := X$;  $R_2 := Y$;  $R_3 := Z$;  $R_7 := X$;  $R_8 := Y$;  $R_9 := Z$;
2: $R_0 := 0$;  $i := n - 2$;  $s := 1$;
3: while $i \geq 0$ do
4:     $k := s == 0?k + 1 : 0$;  $s := d_i == 0?(k \text{ div } 9) : (k \text{ div } 25)$;
5:     $R_{u^*[k,0]} := R_{u^*[k,1]} \cdot R_{u^*[k,2]}$;  $R_{u^*[k,3]} := R_{u^*[k,4]} + R_{u^*[k,5]}$;
6:     $R_{u^*[k,6]} := -R_{u^*[k,6;]}$;  $R_{u^*[k,7]} := R_{u^*[k,8]} + R_{u^*[k,9]}$;
7:     $i := i - s$;
8: return $(R_1, R_2, R_3)$;
```

It is worthy to note that, in terms of field multiplications, Algorithm 5 is as efficient as the unprotected binary double–and–add implementation, namely Algorithm 1. So its complexity is the same as Algorithm 1:

$$(n - 2)[ECDBL] \quad + \quad H(d)[ECADD] \tag{4}$$

$H(d)$ stands for the Hamming weight of d.

The matrix $u^*_{k,l}$ is as follows. For the details why choosing such an array, we refer the interesting readers to [15].

The values of $(u^*_{k,l})_{\substack{0\leq k\leq 25\\ 0<l<9}}$

4	1	1	5	4	4	3	4	4	5
5	3	3	1	1	1	3	1	1	3
5	5	5	1	1	3	3	1	1	3
5	0	5	4	4	5	3	5	2	2
3	3	5	1	1	3	3	1	1	3
2	2	2	2	2	2	4	1	1	3
5	1	2	1	1	5	5	1	1	5
1	4	4	1	1	5	4	1	1	5
2	2	2	2	2	2	3	5	1	5
4	4	5	2	2	4	2	4	4	5
4	9	9	5	1	5	5	5	1	5
1	1	4	5	1	5	5	5	1	5
4	4	9	5	1	5	5	5	1	5

2	2	4	5	1	5	5	5	1	5
4	3	3	5	1	5	5	5	1	5
5	4	7	2	2	5	5	5	1	5
4	3	4	2	2	5	6	6	5	6
4	4	8	6	5	6	4	4	2	4
3	3	9	6	5	6	6	6	5	6
3	3	5	6	5	6	6	6	5	6
6	5	5	6	3	6	3	6	3	6
1	1	6	1	1	4	4	1	1	4
5	5	6	6	1	2	2	6	2	6
1	4	4	1	1	5	6	1	1	6
2	2	5	1	1	6	3	6	1	6
4	4	6	2	2	4	6	6	1	6

3 The SPA-resistant width-w window method

In the former section, we have just studied a window method using NAF for the computation of scalar multiplication, and then the side–channel atomicity. Based on these algorithms, we can now derive an SPA–resistant width–w window method as shown in Algorithm 6. The matrix $u^*_{k,l}$ is the same as in Algorithm 5.

Algorithm 6: Side-channel atomic width-w window method for elliptic curves over F_p

```
    Input: $P = (X, Y, Z), d$, matrix($u^*_{k,l}$), width $w$
    Output: $P_d = dP$
//Pre-computation: Compute $uP$ for $u$ odd and $2 < u < 2^{w-1}$
1:  $P_0 := P$;   $T := 2P$;
2:  for i from 1 to $2^{w-2} - 1$ do
            $P_i := P_{i-1} + T$;
//Main computation
3:  compute $NAF_w(d) := (d_{l-1} \cdots d_1 d_0)$;   //$d_{l-1} \neq 0$
4:  $t := (|d_{l-1}| - 1)/2$;
5:  if $d_{l-1} > 0$ do
            {$R_1 := P_t X$;   $R_2 := P_t Y$;   $R_3 := P_t Z$;}
6:  else do
            {$R_1 := (-P_t)X$;   $R_2 := (-P_t)Y$;   $R_3 := (-P_t)Z$;}
7:  $R_0 := 0$;   $i := l - 2$;   $s := 1$;
8:  while $i \leq 0$ do
9:      $k := s == 0?k + 1 : 0$;
10:     $s := d_i == 0?(k$ div 9$) : (k$ div 25$)$;
11:     $t := (|d_i| - 1)/2$;
12:     if $d_i > 0$ do   //$Q \leftarrow Q + P_t$
```

```
            $\{R_7 := P_tX; \quad R_8 := P_tY; \quad R_9 := P_tZ\}$
13:     if $d_i < 0$ do   $//Q \leftarrow Q - P_t$
            $\{R_7 := (-P_t)X; \quad R_8 := (-P_t)Y; \quad R_9 := (-P_t)Z\}$
14:     else          $//Q \leftarrow 2Q$
            $\{R_7 := X; \quad R_8 := Y; \quad R_9 := Z\}$
15:     $R_{u^*[k,0]} := R_{u^*[k,1]} \cdot R_{u^*[k,2]}; \quad R_{u^*[k,3]} := R_{u^*[k,4]} + R_{u^*[k,5]};$
16:     $R_{u^*[k,6]} := -R_{u^*[k,6;]}; \quad R_{u^*[k,7]} := R_{u^*[k,8]} + R_{u^*[k,9]};$
17:     $i := i - s;$
18:   return $(R_1, R_2, R_3);$
```

Since the underlying side–channel atomicity does not bring in redundant EC–operations, so, in terms of field multiplications, this SPA-resistant algorithm is as efficient as the unprotected width–w window method(i.e. Algorithm 4).

That is,it involves(l is the length of d in NAF)

- $2^{w-2} - 1$ additions plus one doubling in the pre-computation step, and
- $l/(w+1)$ additions plus $l-1$ doublings in the main computation step.

So the overall complexity of Algorithm 6 is as follows:

$$(2^{w-2} - 1 + l/(w+1))[ECADD] \quad + \quad l[ECDBL] \tag{5}$$

From (4), we know that the time complexity of Algorithm 5 is

$$H(d)[ECADD] + (n-2)[ECDBL]$$

where n is the length of d and $H(d)$ is the Hamming weight of d.

Now let't take a closer look at them.

- With proper selected width w(i.e. $3 \leq w \leq 8$),we have $l < n/2$, and then $2^{w-2} - 1 + l/(w+1) < n - 2$.
- $H(d)$ is greater than $l/(w+1)$ in normal sense due to the nature of NAF.

So now we can draw our conclusion that Algorithm 6 is a more efficient algorithm than Algorithm 5, while remaining the same level of security.

4 Parallel scalar multiplication

In [12], Garcia et al. propose a parallel algorithm (we name it here Garcia's parallel algorithm) for the computing of scalar multiplications

(dP), which is especially efficient for the Koblitz curves[4]. Here in our paper, we propose a new parallel scalar multiplication algorithm, which is more compatible with the SPA–resistant width–w window method mentioned above.

Firstly, we divide the binary representation of d into p blocks of $l = \lceil n/p \rceil$ bits each one(this bit scattering pattern is different from the one in [12]). It means we split d into $s_0, s_1, \cdots, s_{p-1}$ as follows:

$$\begin{aligned} s_0 &= (d_{l-1} \cdots d_1 d_0) 2^0 \\ s_1 &= (d_{2l-1} \cdots d_{l+1} d_l) 2^l \\ &\vdots \\ s_{p-1} &= (d_{(p-1)l+l-1} \cdots d_{(p-1)l+1} d_{(p-1)l}) 2^{(p-1)l} \end{aligned} \tag{6}$$

That is, for $i = 0, 1, \cdots, p-1$, we have

$$\begin{aligned} s_i &= (d_{il+l-1} \cdots d_{il+1} d_{il}) 2^{il} = D_i 2^{il} \\ D_i &= (d_{il+l-1} \cdots d_{il+1} d_{il}) \end{aligned} \tag{7}$$

So we get:

$$\begin{aligned} d &= s_0 + s_1 + \cdots + s_{p-1} \\ dP &= \sum_{i=0}^{p-1} s_i P = \sum_{i=0}^{p-1} D_i 2^{il} P \end{aligned} \tag{8}$$

Now, we can compute $s_i P$ in parallel, and then collect and add up all the immediate results to obtain the final result of dP.

Here, in order to obtain fast computation of $2^{il}P$, we adopt the proposed algorithm, namely Itoh's m–repeated Elliptic Doubling Routine, in [8], which is faster than the one in IEEE P1363 draft. As for the computing of $2^l P$, the complexity of the former method is $(8l+2)([m]+[a])$, while that of the latter is $10l[m] + 13l[a]$. For more details see [8]. The proposed parallel scalar multiplication is as Algorithm 7.

Algorithm 7: Parallel scalar multiplication for elliptic curves over F_p

```
    Input: integer $d$, $n$(length of $d$), a base point $P$
           and the processor number $p$
    Output: $dP$
//step 1: the computation of $2^{il}P$
1:  $P_0 := P$;
2:  for j from 1 to $p-1$ do
```

```
       {$P_j$:=Itoh($l, P_{j-1}$)}//Itoh's m–repeated elliptic doubling routine
**//step 2: splitting of d into $D_i$**
3:   divide d into the set $\{D_0, D_1, \cdots, D_{p-1}\}$
**//step 3: parallel computing**
4:   for each $i(0 \le i \le p-1)$ par–do //using p processor elements(PE)
       $Q_i$:=S–window($D_i, P_i$); //S–window denotes Algorithm 6
**//step 4: results collecting**
5:   for $t := 1$ to $logP$ do
6:       for $j := 0$ to $p-1$ step $2^t$ par–do //using $p/2$ PEs at most
           $Q_j := Q_j + Q_{j+2^{t-1}}$;                //the immediate results
7:   return $Q_0$;
```

Now let us take a look at the complexity of Algorithm 7 briefly as follows.

- In step 1, we have to compute Itoh() $p-1$ times, so the time complexity is $(p-1)(8l+2)([m]+[a])$
- In step 2, the splitting can be obtained in a negligible time.
- In step 3, the complexity is $(2^{w-2} - 1 + l/(w+1))[ECADD] + l[ECDBL]$, including the pre-computation time of algorithm 6.
- In step 4, obviously it can be completed in $\lceil logP \rceil$ steps, so its complexity is close to $\lceil logP \rceil$[ECADD].

Thus, the total time complexity of algorithm 7 is as follows:

$$\begin{aligned} T = \quad & (p-1)(8l+2)([m]+[a]) \\ & + (2^{w-2} - 1 + l/(w+1) + \lceil logP \rceil)[ECADD] \\ & + l[ECDBL] \end{aligned} \tag{9}$$

Since $[ECDBL] = 11[a] + 4[m] + 6[s] \approx 11[a] + 10[m]$, we have

$$(8l+2)([m]+[a]) < (4l/5)[ECDBL]$$

Further, we know $l = \lceil n/p \rceil$,then the time complexity of Algorithm 7(i.e. T) satisfy the following inequation:

$$\begin{aligned} T < \quad & (2^{w-2} - 1 + \lceil n/p \rceil/(w+1) + \lceil logP \rceil)[ECADD] \\ & + ((4p/5 + 1/5)\lceil n/p \rceil)[ECDBL] \end{aligned} \tag{10}$$

From [12], we know that, in the best case, the time complexity of Garcia's parallel algorithm is:

$$T_G = (\lceil H(d)/p \rceil + \lceil logP \rceil)[ECADD] + (n-1)[ECDBL] \tag{11}$$

In order to do a comparison between the proposed parallel algorithm and Garcia's parallel algorithm, we compare with each other the coefficients of [ECADD] and [ECDBL] of (10) and (11) respectively.

Firstly, we consider the case of [ECADD]. With proper selected w (i.e. $3 \leq w \leq 8$), the value of 2^{w-2} is pretty small in contrast to n, and thus is negligible. As a result of the form NAF, we know $H(d)$ is $n/(w+1)$ on average. So in this case, the time consumptions are approximately the same.

Secondly, it is the case of [ECDBL]. Due to the fact that the processor number p is greater than 1, it is obviously that $(4p/5+1/5)\lceil n/p \rceil < n-1$ holds.

So we safely come to this point that $T < T_G$ holds in a reasonable way. That is, our proposed parallel algorithm is a more efficient one.

Further more, if the base point P and the length of scalar d are available in advance, step 1 of Algorithm 7 can be pre–computed. By this, the algorithm can be accelerated, and achieve an even better performance.

Further, even integrated with Algorithm 6, Garcia's parallel algorithm cannot achieve a better performance than Algorithm 7. This is due to the fact that, the bit scattering pattern it adopted could not fit well. That is, the number of the computing of $2^i P$(step 1, Algorithm 7) is much bigger than that in our proposed algorithm, because here i is from 1 to $\lceil n/p \rceil$ (greater than p in Algorithm 7).

It is worthy to note that, as the underlying blocks of Algorithm 7 is secure against SPA, we have proposed an efficient and SPA–resistant parallel scalar multiplication algorithm for elliptic curves over the prime field F_p.

5 Conclusion

In this paper, we firstly give a list of backgrounds, which include side–channel analysis etc., and then we utilize the window method and side–channel atomicity to compose a SPA–resistant width–w window method, which is both efficient and secure against simple power attack. After that, we propose a new parallel scalar multiplication algorithm, which is integrated with the SPA–resistant width–w window method and Itoh's m–repeated doubling routine. The proposed parallel algorithm is efficient and SPA–resistant.

To make our parallel algorithm be DPA–resistant, one only need to add some randomization into it. There are several ways to achieve this, see [14] for details.

References

[1] G. Reitwiesner, *Binary arithmetic*, Advances in Computers, 1, pp.231–308, 1960

[2] V. Miller, *Use of elliptic curves in cryptography*, Advances in Cryptography: proceedings of Crypto'85, LNCS 218, pp.417-426, New York: Springer-Verlag, 1986

[3] N. Koblitz, *Elliptic curve cryptosystems*, Mathematics of Computation, 48, pp.203-209, 1987

[4] N. Koblitz, *CM–curves with good cryptographic properties* , Advances in Cryptology-CRYPTO'91, LNCS 576, pp.279-287, 1992

[5] P.C. Kocher, *Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems*, In Advances in Cryptology - CRYPTO'96, vol. 1109 of Lecture Notes in Computer Science, pages 104-113. Springer-Verlag, 1996

[6] P.C. Kocher,J. Jaffe, and B. Jun, *Differential power analysis*, In Advances in Cryptology - CRYPTO'99, vol. 1666 of Lecture Notes in Computer Science, pages 388-397. Springer-Verlag, 1999

[7] I. Blake, G. Seroussi, and N. Smart, *Elliptic Curves in Cryptography*, Cambridge University Press, 1999

[8] K. Itoh, M. Takenaka, N. Torii, S. Temma and Y. Kurihara, *Fast Implementation of Public-Key Cryptography on a DSP TMS320C6201*, Proceedings of the First International Workshop on Cryptographic Hardware and Embedded Systems, pp.61-72, Springer Verlag, 1999

[9] J. Solinas, *Improved Algorithms for Arithmetic on Anomalous Binary Curves*, CACR(Univ. of Waterloo) Technical Report, CORR 99-46, http://www.cacr.math.uwaterloo.ca/techreports/1999/corr99-46.pdf, 1999

[10] J. Solinas, *Efficient Arithmetic on Koblitz Curves*, in Designs,Codes and Cryptography, volume 19, issue 2-3, pp. 195-249, 2000

[11] J. Lopez and R. Dahab, *An Overview of Elliptic Curve Cryptography*, http://citeseer.ist.psu.edu/lop00overview.html, 2000

[12] J. Garcia, and R. Carcia, *Parallel algorithm for multiplication on elliptic curves*, Proceedings of the ENC'01, September 2001

[13] T. Izu, B. Moller, and T. Takagi, *Improved Elliptic Curve Multiplication Methods Resistant against Side Channel Attacks*, Progree in Cryptology-INDOCRYPT 2002, Springer-Verlag LNCS 2551, pp. 296-313, 2002

[14] M. Joye, *Elliptic Curve Cryptography and Side Channel Attacks*, http://ca.itsc.ruhr-uni-bochum.de/hgi/smaca/bochum-joye.pdf, 2003

[15] B. Chevallier-Mames, M. Ciet and M. Joye, *Low-Cost Solutions for Preventing Simple Side-Channel Analysis: Side-Channel Atomicity*, IEEE Trans. on Computers, 53(6):760-768, 2004

[16] D. Hankerson, A. Menezes and S. Vanstone, *Guide to Elliptic Curve Cryptography*, Springer-Verlag, 2004

[17] P. Mishra, *Pipelined Computation of Scalar Multiplication in Elliptic Curve Cryptosystems*, CHES 2004, LNCS 3156, pp. 328-342, 2004, Springer-Verlag

[18] P. Mishra, *Scalar Multiplication in Elliptic Curve Cryptosystems: Pipelining with Pre-computation*, 2004