Parallel scalar multiplication on general elliptic curves over $\mathbb{F}_p$ hedged against Non-Differential Side-Channel Attacks

Wieland Fischer[1], Christophe Giraud[2],
Erik Woodward Knudsen[2] and Jean-Pierre Seifert[1]

[1] Infineon Technologies, St. Martin-Straße 76, 81541 Munich, Germany.
[2] Oberthur Card Systems, 25, rue Auguste Blanche, 92800 Puteaux, France.

Abstract. For speeding up elliptic curve scalar multiplication and making it secure against side-channel attacks such as timing or power analysis, various methods have been proposed using specifically chosen elliptic curves. We show that both goals can be achieved simultaneously even for conventional elliptic curves over $\mathbb{F}_p$. This result is shown via two facts. First, we recall the known fact that every elliptic curve over $\mathbb{F}_p$ admits a scalar multiplication via a (Montgomery ladder) Lucas chain. As such chains are known to be resistant against timing- and simple power/electromagnetic radiation analysis attacks, the security of our scalar multiplication against timing and simple power/electromagnetic radiation analysis follows. Second, we show how to parallelize the 19 multiplications within the resulting "double" and "add" formulas of the Lucas chain for the scalar multiplication. This parallelism together with the Lucas chain results in 10 parallel field multiplications per bit of the scalar.
Finally, we also report on a concrete successful implementation of the above mentioned scalar multiplication algorithm on a very recently developed and commercially available coprocessor for smart cards.

Keywords: Elliptic Curves, Montgomery ladder, Power Analysis, Timing Analysis, Electro Magnetic Radiation Analysis, Efficient and Parallel Implementation, Scalar Multiplication.

1 Introduction

It is clear that one of the major application fields of elliptic curve cryptography (as described in [4, 13, 20]) are smart cards, mobile phones and PDAs. Their strong limitations concerning chip area, power consumption and performance can be remedied by the use of elliptic curve cryptography in a sophisticated way, see, e.g. [1, 21].

Unfortunately, real physical implementations of elliptic curve cryptography within smart cards suffer from so called side-channel attacks (also known as information leakage attacks), cf. [6, 7, 24, 27, 29]. These are attacks on implementations of cryptosystems which use observations like timings [16], power consumptions [17] or electromagnetic radiation [26] in order to obtain secret information, that is originally supposed to be stored safely in the tamper-resistant device.

Not surprisingly, various proposals have been made to secure the elliptic curve scalar multiplication against different side-channel attacks. Although we will not discuss the details of these methods we will briefly summarize the existing proposals. The first who addressed the SPA and DPA problem for elliptic curves was [6], which suggested several randomization techniques to defeat differential power analysis. Later, [3] investigated the elliptic curve scalar multiplication security concerning Koblitz curves. More importantly, [29] rediscovered the so called Montgomery ladder to defeat side-channel attacks on the elliptic curve scalar multiplication — a possibility which was already pointed out earlier by [28]. Originally, the Montgomery ladder was used by [23] and [19] to accelerate the elliptic curve scalar multiplication of certain curves. Furthermore, [29] found some security flaws in the methods of [6, 19, 28] and also proposed a (TA, SPA and DPA) secure variant of the Montgomery ladder. Finally, on CHES 2001 several other methods have been suggested by [14, 15, 18] to defeat side-channel attacks on elliptic curves. A common disadvantage of the above methods is that each of them requires specifically selected curves which often constitutes only a small class of all curves and that these are most often not included in the standards [2, 9, 25].

As said above and already pointed out by [3, 6, 15, 18, 24, 29, 31] the immunity against differential side-channel attacks on elliptic curve scalar multiplication can easily be achieved by virtue of the rich algebraic structure of elliptic curves, e.g., point blinding using the redundancy of a projective coordinate representation. Therefore, this article concentrates on non-differential side-channel attacks which are described in the next section.

Also, only very recently, [31] addressed the idea of parallelizing the elliptic curve scalar multiplication for the Hessian form of an elliptic curve. But in contrast to [31] our multiplication method has several benefits. First of all, we address simultaneously the side-channel attack problem and the parallelization problem. Second, our method works for an arbitrary curve in Weierstrass form as specified in [2, 9, 25], whereas the Hessian form constitutes a limited class. Thus, our paper proposes an alternative countermeasure approach without a limitation to specifically chosen curves. Third, our method works on a commercially available standard smart card system in contrast to [31].

The rest of the paper is organized as follows. In the next section we briefly summarize non-differential information leakage attacks and how they can reveal the secret keys of an elliptic curve cryptosystem. Moreover, we explain the principles of how to overcome this information leakage via the above mentioned Montgomery ladder. Hereafter, we shortly present in the next section the necessary details concerning the arithmetic of elliptic curves. Next, we turn our attention to the so called Montgomery ladder or Lucas chain. Although it is already known (cf. [8, 12]) that every elliptic curve over $\mathbb{F}_p$ admits a scalar multiplication via a Montgomery ladder, we systematically derive this result from the classical addition formulas for affine coordinates and also for the more practical projective coordinates. We feel that it is necessary to recall this derivation as it helps to understand the presentation of our parallel algorithm for the Montgomery ladder. We also discuss in another section a real implementation of our algorithm on a parallel coprocessor for a smart card.

2 Immunity against Non-Differential Side-Channel Attacks

In elliptic curve cryptosystems, a particular target for side-channel attacks are the algorithms used for the elliptic curve scalar multiplication. Within the aforesaid multiplication one is given a positive integer k, a point P on a curve $\mathcal{C}$ over a finite field $\mathbb{F}_q$ and the task is to add k times the point P to itself, which is usually denoted as $k \cdot P$, as the elliptic curves are known to be abelian additive groups (cf. [30]) with respect to the addition of points. Most often, the point

P is public and can therefore be chosen by an attacker, while the scalar k is secret and thus of special interest for the attacker.

However, for reasons of exactness one has to distinguish between two classes of side-channel attacks: differential (DPA and DEMA) and non-differential side-channel attacks (TA, SPA and SEMA).

An attack relying on a differential analysis guesses unknown (hardware or software) internal bits and then correlates this guessing over a large number of runs. As already said above, these attacks are easily prevented by the formerly mentioned and well understood randomization techniques concerning elliptic curves.

Non-differential attacks are conceptually much simpler and they are harder to safe guard against without a performance loss. To understand our defence against non-differential attacks for the elliptic curve point multiplication we recall the binary ladder (also known as classical double and algorithm) for point multiplication. This is the standard way of computing the scalar multiplication.

Input: An integer $k = (k_{n-1}, \ldots, k_0)$ and a point P on the curve $\mathcal{C}$.
Output: kP.

1. $Q := \mathcal{O}$.
2. For i from $n-1$ down to 0 do
   $Q := 2Q$
   if $k_i = 1$ then
     $Q := Q + P$.
3. Return Q.

Fig. 1. The binary ladder or the double and add algorithm.

As one sees in the above algorithm, an attacker can attempt to determine the bits of k by seeing how the program behaves at the *if-branch*. Namely, whereas the point doubling is always executed, the point addition is executed conditionally on the ith bit of k. Thus, the attacker is able to determine from a timing analysis, a simple power or a electromagnetic radiation analysis whether the *if-branch* was executed or not.

The most common idea is to make point addition and point doubling indistinguishable by adding dummy code to balance the difference between addition and doubling even. But one has to recall that this has to be done down to the level of the underlying hardware. However, this is not a trivial task to do in practice and also leads to a performance loss. Thus, it is not a recommendable countermeasure.

In contrast to this naive way, we will use the formerly mentioned Montgomery ladder, see, e.g., [23, 8]. The Montgomery ladder solves the problem, as it leads to a uniform execution pattern, as it always performs an addition and a doubling independently of the scalar k. It exploits the fact that if the difference of two points is known, it is easier to compute their sum. The Montgomery ladder was originally introduced to accelerate the scalar multiplication on a certain restricted class of curves defined over $\mathbb{F}_p$. Note that there is also a version of this algorithm for arbitrary curves over fields of characteristic two [19]. But this algorithm for an arbitrary curve over a field of characteristic two doesn't work for curves over fields $\mathbb{F}_p$ with p greater than 3. This is one aim of our paper. As with the double and add algorithm, it can be represented either in affine or projective coordinates.

The Montgomery ladder works as follows. Let k be a positive integer and $(k_{n-1}, \ldots, k_0)$ its binary representation where we may assume that $k_{n-1} = 1$. To compute kP we start with the pair $(P, 2P)$. At the beginning of each step i we have the pair $(P_1, P_2) = (mP, (m+1)P)$ where $m = (k_{n-1}, \ldots, k_{n-1-i})$ and at the end we eventually have $(kP,\ (k+1)P)$.

Input: An integer $k \geq 1$ and a point P on the curve $\mathcal{C}$.
Output: kP.

1. $P_1 \leftarrow P$ and $P_2 \leftarrow 2P$.
2. For i from $n-2$ down to 0 do
   if $k_i = 1$ then
      $P_1 \leftarrow P_1 + P_2$ and $P_2 \leftarrow 2P_2$
   else
      $P_2 \leftarrow P_1 + P_2$ and $P_1 \leftarrow 2P_1$.
3. Return P_1.

Fig. 2. The Montgomery Ladder.

First, observe that the above algorithm clearly is resistant to non-differential side-channel attacks (cf. [28]). Second, there is an obvious way to parallelize each of the branches within step 2. However, on real arithmetic coprocessors of smart cards this parallelization is not practical for elliptic curves, as the coprocessors usually have only a limited number of registers to store intermediate results. Nevertheless, for other groups where the basic operation is simpler (e.g. RSA) this kind of parallelization might be of some particular value.

3 Preliminaries

We will now shortly present the necessary facts which are needed in the body of our paper. For a thorough introduction to the arithmetic of elliptic curves we refer to [30].

For p a prime greater than 3, the (short) Weierstrass form of an elliptic curve $\mathcal{C}$ defined over $\mathbb{F}_p$ is given by the equation

$$y^2 = x^3 + ax + b. \tag{1}$$

where $a, b \in \mathbb{F}_p$ are such that $4a^3 + 27b^2 \not\equiv 0 \bmod p$. Any elliptic curve defined over $\mathbb{F}_p$ can be expressed in this form. Also note that the $\mathbb{F}_p$-rational points on an elliptic curve together with the point at infinity $\mathcal{O}$ form an abelian group with the neutral element $\mathcal{O}$. For points $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$ different from $\mathcal{O}$, their sum $P_3 = (x_3, y_3)$ is defined by the following equations. If $y_1 = -y_2$ then $P_3 = \mathcal{O}$, otherwise

$$\begin{cases} x_3 = \lambda^2 - x_1 - x_2, \\ y_3 = \lambda(x_1 - x_3) - y_1, \end{cases}$$

where

$$\lambda = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1}, \text{ if } P_1 \neq P_2, \\ \\ \frac{3x_1^2 + a}{2y_1}, \text{ if } P_1 = P_2. \end{cases}$$

4 Presentation of the Algorithm

In the following we will use some ideas from [8] in order to show that every elliptic curve over $\mathbb{F}_p$ admits a scalar multiplication via a Montgomery ladder.

For a point P on the curve $\mathcal{C}$ we write $P = (x_P, y_P) = (X_P : Y_P : Z_P) \in \mathbb{A}^2 = \mathbb{P}^2 - \mathbb{P}^1$ in affine resp. projective coordinates. For these coordinates we have $x_P \cdot Z_P = X_P$ and $y_P \cdot Z_P = Y_P$. For two given distinct points P and Q, different from $\mathcal{O}$, our task is the computation of the pair

$$(P', Q') \ := \ (P + Q, 2Q) \tag{2}$$

from the pair (P, Q) with the additional notion of the point $D := P - Q$ (or $D := Q - P$, but they differ only in the sign of their y-coordinate, which is not going to be used). Recall that in our case D equals the base point of the curve (or its opposite).

The standard formulas for addition and doubling on elliptic curves from above yield the three formulas

$$x_{P'} = \left(\frac{y_P - y_Q}{x_P - x_Q}\right)^2 - x_P - x_Q \tag{3}$$

for the sum of P and Q,

$$x_D = \left(\frac{y_P + y_Q}{x_P - x_Q}\right)^2 - x_P - x_Q \tag{4}$$

for the difference of P and Q and

$$x_{Q'} = \left(\frac{3x_Q^2 + a}{2y_Q}\right)^2 - 2x_Q \tag{5}$$

for the double of Q. Adding (3) and (4) and substituting y_P^2 and y_Q^2 by virtue of (1) we get

$$(x_{P'} + x_D)(x_P - x_Q)^2 = 2(x_P + x_Q)(x_P x_Q + a) + 4b. \tag{6}$$

Substituting y_P^2 via (1) we get from (5)

$$4x_{Q'}({x_Q}^3 + ax_Q + b) = ({x_Q}^2 - a)^2 - 8bx_Q. \tag{7}$$

Now we substitute X_P/Z_P for x_P and X_Q/Z_Q for x_Q and transform (6) into

$$\begin{aligned} X_{P'}(X_P Z_Q - X_Q Z_P)^2 = Z_{P'}\Big(&2(X_P Z_Q + X_Q Z_P)(X_P X_Q + aZ_P Z_Q) \\ &+4bZ_P^2 Z_Q^2 - x_D(X_P Z_Q - X_Q Z_P)^2\Big) \end{aligned}$$

and (7) into

$$X_{Q'}4(X_Q Z_Q(X_Q^2 + aZ_Q^2) + bZ_Q^4) = Z_{Q'}\left((X_Q^2 - aZ_Q^2)^2 - 8bX_Q Z_Q^3\right).$$

Therefore, we can define the "double and add" formulas

$$\begin{cases} X_{P'} = 2(X_P Z_Q + X_Q Z_P)(X_P X_Q + aZ_P Z_Q) \\ \qquad\quad +4b{Z_P}^2{Z_Q}^2 - x_D(X_P Z_Q - X_Q Z_P)^2 \\ Z_{P'} = (X_P Z_Q - X_Q Z_P)^2 \\ X_{Q'} = ({X_Q}^2 - a{Z_Q}^2)^2 - 8bX_Q{Z_Q}^3 \\ Z_{Q'} = 4(X_Q Z_Q({X_Q}^2 + a{Z_Q}^2) + b{Z_Q}^4), \end{cases} \tag{8}$$

always ignoring the Y-coordinates.

We now present an algorithm for $(P, Q) \mapsto (P', Q')$ which uses 8 registers R0, ..., R7 and is intended to work on a parallel computing device with unconstrained access to all registers.

Input: (X_P, Z_P, X_Q, Z_Q)
Output: $(X_{P'}, Z_{P'}, X_{Q'}, Z_{Q'})$
$R0 \leftarrow X_P, R1 \leftarrow Z_P, R2 \leftarrow X_Q, R3 \leftarrow Z_Q$

$R6 \leftarrow R2 \cdot R1$	(1)	$R7 \leftarrow R3 \cdot R0$	(2)
$R4 \leftarrow R7 + R6$	(3)	$R5 \leftarrow R7 - R6$	(4)
$R5 \leftarrow R5 \cdot R5$	(5)	$R7 \leftarrow R1 \cdot R3$	(6)
$R1 \leftarrow a \cdot R7$	(7)	$R6 \leftarrow R7 \cdot R7$	(8)
$R0 \leftarrow R0 \cdot R2$	(9)	$R6 \leftarrow b \cdot R6$	(10)
$R0 \leftarrow R0 + R1$	(11)	$R6 \leftarrow R6 + R6$	(12)
$R0 \leftarrow R0 \cdot R4$	(13)	$R1 \leftarrow x_D \cdot R5$	(14)
$R4 \leftarrow R0 + R6$	(15)		
$R4 \leftarrow R4 + R4$	(16)	$R6 \leftarrow R2 + R2$	(17)
$R4 \leftarrow R4 - R1$	(18)	$R7 \leftarrow R3 + R3$	(19)
$R0 \leftarrow R6 \cdot R7$	(20)	$R1 \leftarrow R3 \cdot R3$	(21)
$R2 \leftarrow R2 \cdot R2$	(22)	$R3 \leftarrow a \cdot R1$	(23)
$R6 \leftarrow R2 - R3$	(24)	$R7 \leftarrow R2 + R3$	(25)
$R1 \leftarrow R1 + R1$	(26)		
$R2 \leftarrow b \cdot R1$	(27)	$R7 \leftarrow R7 \cdot R0$	(28)
$R1 \leftarrow R2 \cdot R1$	(29)	$R0 \leftarrow R0 \cdot R2$	(30)
$R6 \leftarrow R6 \cdot R6$	(31)		
$R6 \leftarrow R6 - R0$	(32)	$R7 \leftarrow R7 + R1$	(33)

$X_{P'} \leftarrow R4, Z_{P'} \leftarrow R5, X_{Q'} \leftarrow R6, Z_{Q'} \leftarrow R7$

Fig. 3. Parallel computing of $(P', Q') = (P + Q, 2Q)$.

The above algorithm — consisting of 19 multiplications and 14 additions — needs the time of 10 multiplications and 8 additions on a parallel architecture. On an architecture with only one computing device, one would have to do all 33 steps one by one.

After the scalar multiplication kP we have the projective X-coordinate and Z-coordinate of $kP = (X_{kP} : \ Y_{kP} : \ Z_{kP})$. To obtain the affine coordinates of kP we use the transformation

$$kP = (X_{kP},\ Y_{kP},\ Z_{kP}) \mapsto kP = (X_{kP}/Z_{kP},\ Y_{kP}/Z_{kP}) = (x_{kP},\ y_{kP}).$$

Similarly we have the affine x-coordinate of

$$(k+1)P = (X_{(k+1)P}/Z_{(k+1)P},\ Y_{(k+1)P}/Z_{(k+1)P})$$

and the affine coordinates of the point $P = (x_P,\ y_P)$. So, in order to obtain the affine y-coordinate of the point kP we have the following formula if we substitute y_{kP}^2 and y_P^2 via (1) into (3), applied to the equation $(k+1)P = kP + P$

$$2y_P y_{kP} = -(x_P - x_{kP})^2 x_{(k+1)P} + (a + x_P^2)x_{kP} + (a + {x_{kP}}^2)x_P + 2b. \quad (9)$$

So, if we substitute x_{kP} by X_{kP}/Z_{kP} and $x_{(k+1)P}$ by $X_{(k+1)P}/Z_{(k+1)P}$, we obtain (now omitting most of the Ps in the subscripts)

$$y_k = \frac{2b{Z_k}^2 Z_{k+1} + Z_{k+1}(x_P X_k + aZ_k)(X_k + x_P Z_k) - X_{k+1}(X_k - x_P Z_k)^2}{2y_P Z_k^2 Z_{k+1}}. \quad (10)$$

5 Implementation performance

For the implementation of our algorithm we have chosen Infineons 8-bit chipcard controller SLE66P [10] equipped with the recently developed crypto coprocessor CRYPTO2000 [11] running at 33MHz. The CPU of the SLE66P is an enhanced 8051 compatible architecture.

The new CRYPTO2000 is an arithmetic coprocessor dedicated to modular arithmetic as used in various current cryptographic standard applications. It is an amalgamation of an optimized 2K RSA coprocessor and an optimized 1/2K ECC coprocessor. In the RSA mode the accumulator based machine provides one calculation unit of 2K bits and two registers for temporary results of the same length. In the ECC mode the machine is divided in two identical parallel calculation units. Each of the two registers is divided into four 1/2K registers, resulting in eight temporary result registers altogether.

We have chosen a standard curve from [9] over $\mathbb{F}_p$ with 162 bits for the prime p, P a generating point of this group and a scalar k of size 158 bits. The following table summarizes our timing results for a straightforward implementation which is not optimized concerning performance.

Operation	Comp. the proj. X-coordinate and Z-coordinate of kP	Comp. the affine x-coordinate of kP	Comp. the affine coordinates of kP
Time in ms	11.5	13.7	16

Fig. 4. Timings to compute the affine coordinates of kP.

6 Conclusion

We have presented a method for elliptic curve point multiplication that provides immunity against non-differential side-channel attacks. The algorithm exploits the so called Montgomery ladder to ensure that point doublings and point additions occur in a uniform pattern. No dummy additions are required. The method can easily be parallelized in a practical way, thus resulting in a performance superior to all existing methods to provide immunity for elliptic curve point multiplication against non-differential side-channel attacks. In contrast to other methods which are only applicable to special curves our algorithm works for an arbitrary elliptic curve. Therefore, our method is the perfect choice for a fast, secure and practical ECC implementation on a parallel computing device.

7 Acknowledgments

Wieland Fischer and Jean-Pierre Seifert would like to thank Norbert Janssen and Holger Sedlak for lots of valuable discussions concerning the development of the CRYPTO2000 and as well Christophe

Giraud and Erik Knudsen for acting as β-testers of the CRYPTO2000 simulator.

References

1. G. B. Agnew, R. C. Mullin and S. A. Vanstone, *An implementation of elliptic curve cryptosystems over* $\mathbb{F}_{2^{155}}$, IEEE Journal on Selected Areas in Communications, vol. 11, pp. 804-813, 1993.
2. ANSI, *Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Standard (ECDSA)*, ANSI X9.62, American National Standard Institute, 1999.
3. A. Anwar Hasan, *Power analysis attacks and algorithmic approaches to their countermeasures for Koblitz cryptosystems*, CHES 2000, Springer LNCS, pp. 93-108, 2000.
4. I. Blake, G. Seroussi and N. Smart, *Elliptic Curves in Cryptography*, Cambridge University Press, 1999.
5. H. Cohen, T. Ono and A. Miyaji, *Efficient elliptic curve exponentiation using mixed coordinates*, ASIACRYPT '98, Springer LNCS, pp. 51-65, 1998.
6. J.-S. Coron, *Resistance against differential power analysis for elliptic curve cryptosystems*, CHES '99, Springer LNCS, pp. 292-302, 1999.
7. J.-S. Coron, P. Kocher and D. Naccache, *Statistics and secret leakage*, Proc. of Financial Cryptography '00, pp. ?-?, 2000.
8. R. Crandall, C. Pomerance, *Prime Numbers*, Springer-Verlag, 2001.
9. IEEE, *IEEE Standard Specifications for Public Cryptography*, IEEE Std 1363-2000, IEEE Computer Society, 2000.
10. Infineon Technologies AG, *SLE66P Confidential Data Book*, München, 2000.
11. Infineon Technologies AG, *Crypto2000 Architecture Specification*, München, 2001.
12. T. Izu and T. Takagi, *Fast Parallel Elliptic Curve Multiplication Resistant against Side Channel Attacks*, to appear in International Workshop on Practice and Theory of Public Key Cryptography, Springer LNCS, pp. ?-?, 2002.
13. D. Johnson and A. J. Menezes, *The Elliptic Curve Digital Signature Algorithm (ECDSA)*, International Journal of Information Security, vol. 1, pp. 36-63, 2001.
14. M. Joye and J.-J. Quisquater, *Hessian elliptic curve cryptosystems*, CHES 2001, Springer LNCS, pp. 412-420, 2001.
15. M. Joye and C. Tymen, *Protections against Differential Analysis for Elliptic Curve Cryptography*, CHES 2001, Springer LNCS, pp. 386-400, 2001.
16. P. C. Kocher, *Timing attacks on implementations of DH, RSA, DSS and other systems*, CRYPTO '96, Springer LNCS, pp. 104-113, 1996.
17. P. C. Kocher, J. Jaffe and B. Jun *Differential power analysis*, CRYPTO '99, Springer LNCS, pp. 388-397, 1999.
18. P.-Y. Liardet and N. P. Smart, *Preventing SPA/DPA in ECC systems using the Jacobi form*, CHES 2001, Springer LNCS, pp. 401-411, 2001.
19. J. López and R. Dahab, *Fast Multiplication on Elliptic Curves over $GF(2^m)$ without Precomputation*, CHES '99, Springer LNCS, pp. 316-327, 1999.
20. A. J. Menezes, *Elliptic Curve Public Key Cryptosystemes*, Kluwer Academic Publishers, 1993.
21. A. J. Menezes and S. A. Vanstone, *Elliptic Curve Cryptosystems and their Implementation* J. of Cryptology, pp. 209-224, 1993.

22. A. Miyaji and H. Cohen, *Efficient elliptic curve exponentiations*, ICICS '97, Springer LNCS, pp. 282-290, 1997.
23. P. L. Montgomery, *Speeding the Pollard and Ellitpic Curve Methods of Factorization*, Mathematics of Computation, vol. 48, pp. 243-264, 1987.
24. B. Möller, *Securing Elliptic Curve Point Multiplication against side-channel attacks*, ISC '01, Springer LNCS, pp. ?-?, 2001.
25. NIST, *Digital Signature Standard (DSS)*, FIPS PUB 186-2, National Institute of Standards and Technology, 2000.
26. J.-J. Quisquater and D. Samyde, *ElectroMagnetic Analysis (EMA): Measures and Countermeasures for Smart Cards*, E-SMART 2001, Springer LNCS, pp. 200-210, 2001.
27. T. Römer and J.-P. Seifert, *Information leakage attacks against Smart Card implementations of the ECDSA*, E-SMART 2001, Springer LNCS, pp. 211-219, 2001.
28. K. Okeya, H. Kurumatani and K. Sakaurai, *Elliptic Curves with the Montgomery-Form and their Cryptographic Applications*, PKC 2000, Springer LNCS, pp. 238-257, 2000.
29. K. Okeya and K. Sakaurai, *Power Analysis breaks Elliptic Curve cryptosystems even secure against the timing attacks*, INDOCRYPT 2000, Springer LNCS, pp. 178-190, 2000.
30. J. H. Silverman, *The Arithmetic of Elliptic Curves*, Springer-Verlag, 1986.
31. N. P. Smart, *The Hessian form of an elliptic curve*, CHES 2001, Springer LNCS, pp. 121-128, 2001.