A Cyclic Window Algorithm for ECC Defined over Extension Fields

Kazumaro Aoki[1*], Fumitaka Hoshino[2], and Tetsutaro Kobayashi[2]

[1] NTT Communications
[2] NTT Information Sharing Platform Laboratories, NTT Corporation
{maro,fhoshino,kotetsu}@isl.ntt.co.jp

Abstract. This paper presents a new sliding window algorithm that is well-suited to an elliptic curve defined over an extension field for which the Frobenius map can be computed quickly, e.g., optimal extension field. The algorithm reduces elliptic curve group operations by approximately 15% for scalar multiplications for a practically used curve in comparison with Lim-Hwang's results presented at PKC2000, the fastest previously reported. The algorithm was implemented on computers. As a result, scalar multiplication can be accomplished in 573μs, 595μs, and 254μs on Pentium II (450 MHz), 21164A (500 MHz), and 21264 (500 MHz) computers, respectively.

1 Introduction

Many studies have been conducted on fast exponentiation algorithms, since many public key cryptosystems require exponentiation g^e [1, Sect. 8]. If the base or exponent cannot be fixed, the sliding window exponentiation [1, 14.85 Algorithm in p. 616] is one of the most efficient algorithms for computing exponentiations.

Recently, elliptic curve cryptosystems have been the focus of much attention, since there are many advantages, for example, a short key length and fast computation speed. In particular, the use of an optimal extension field (OEF) [2] for software implementation has determined that an elliptic curve cryptosystem is faster than a public key cryptosystem based on modular exponentiations.

The algorithms for an exponentiation can be used for scalar multiplications in group operations defined over an elliptic curve. Moreover, if an extension field is used for an elliptic curve, another technique called the base-ϕ expansion [3] can be employed. Until now, several studies on base-ϕ expansion have been conducted including application to OEF [4], but these studies focused on expanding a scalar representation to a base-ϕ representation and the results of the expansion are only analogous to modular exponentiation algorithms.

Solinas presented base-ϕ expansion with a window algorithm for an elliptic curve [5]. His algorithm was considered only for a field with characteristic two, and he used a (sliding) window algorithm. While, Lim and Hwang presented the

[*] This work was done while the author was in NTT Information Sharing Platform Laboratories.

Lim-Lee algorithm [6] application to an elliptic curve defined over OEF [7,8]. They used a more sophisticated algorithm to compute a scalar multiplication using base-ϕ representation, but they did not use any special property derived from elliptic curve properties. Moreover, Tsuruoka and Koyama used optimal addition sequences for this scenario [9]. After finishing constructing the addition sequence, their algorithm performs fast, but computing the addition sequence requires long time for small extension fields.

This paper focuses on a mixture of the sliding window and base-ϕ expansion. Computing Frobenius map ϕ is very fast in a suitably-represented extension field. This property ensures that we can smoothly "slide" in the other direction for computing a scalar multiplication.

2 Preliminaries

2.1 Elliptic Curve

An elliptic curve cryptosystem consists of group arithmetics defined in an elliptic curve over a finite field. The number of field operations follows the representation of a point on an elliptic curve. We can roughly classify the representation into two groups: one is the projective system which does not require field inversion, and the other is the affine system which requires field inversions. For more details, Cohen et al. [10] presents a discussion on the advantages and disadvantages of these systems.

Below is an example of Jacobian coordinates, one of the projective systems. Using a and b as a parameters, a non-supersingular elliptic curve with a characteristic greater than 3 is defined as

$$Y^2 = X^3 + aXZ^4 + bZ^6 \quad (4a^3 + 27b^2 \neq 0) \ . \tag{1}$$

We can define the group operations for the rational points on the curve.

2.2 Previous Studies on OEF

This section provides a brief overview of the history of the implementations of scalar multiplication on an elliptic curve over OEF.

Bailey and Paar proposed OEFs [2]. An OEF can be represented as $\mathrm{GF}(p)[x]/(f(x))$, where $p = 2^n - c$ ($\log_2 c \leq \frac{1}{2}n$) and $f(x) = x^m - \omega$. (Note that n is the size of p and m is the extension degree.) An OEF is very well suited to software implementation because the conditions,

$$\begin{aligned} 2^n &\equiv c \quad \pmod{p} \\ x^m &\equiv \omega \quad \pmod{f(x)} \ , \end{aligned}$$

can be used to reduce the complexity of field multiplications. They showed that their implementations are significantly faster than the implementations previously reported.

Kobayashi et al. applied the base-ϕ expansion technique [3] to scalar multiplication in $E(\mathrm{GF}(p^m))/\,\mathrm{GF}(p)$ [4]. Similar to an elliptic curve over $\mathrm{GF}(2^m)$, which we call a binary field hereafter, the computational cost of Frobenius endomorphism ϕ, is very low. This condition made their implementations about twice as fast as that reported by Bailey and Paar.

Lim and Hwang presented implementations that do not use any epoch-making techniques; however, their accumulation of minor techniques increased the speed of their implementation by approximately two-fold [7,8].

2.3 Window Algorithms

The window algorithm is an algorithm that computes kP using online precomputation. A brief description of the algorithm is given below.

Step 1: Compute $Q_i \leftarrow iP$ for $(2 \le i < 2^w)$, and let $Q_0 = \mathcal{O}$ and $Q_1 = P$.

Step 2: Using c_i that satisfies $k = \sum_{i=0}^{\lfloor(\log_2 k)/w\rfloor} 2^{wi}c_i \quad (0 \le c_i < 2^w)$, compute

$$kP = \sum_{i=0}^{\lfloor(\log_2 k)/w\rfloor} 2^{wi}Q_{c_i} .$$

It accelerates scalar multiplication in comparison to the binary algorithm [1, 14.76 and 14.79 Algorithms in pp. 614–615].

Moreover, the sliding window algorithm [1, 14.85 Algorithm in p. 616] was proposed to improve the window algorithm. By using c_i and w_i satisfying $k = \sum_{i=0}^{\lfloor(\log_2 k)/w\rfloor} 2^{w_i}c_i$, compute $kP = \sum_{i=0}^{\lfloor(\log_2 k)/w\rfloor} 2^{w_i}Q_{c_i}$, where $w_i \ge w_{i-1} + w$.

This paper applies this approach to the base-ϕ expansion algorithm, and proposes an improved version of the sliding window algorithm called the *cyclic window algorithm* to compute $\sum_{i=0}^{m-1} c_i\phi^i P$.

3 Cyclic Window Algorithm

This section proposes the *cyclic window algorithm* and analyzes the algorithm. The cyclic window algorithm computes scalar multiplication kP.

3.1 Notations

This section defines the notations used in the following sections that have not appeared in the previous sections.

Definition 1. *Let $\overline{x}$ be the complementation of all elements in binary vector x. That is*

$$\overline{[x_0, x_1, \ldots, x_{m-1}]} := [\overline{x_0}, \overline{x_1}, \ldots, \overline{x_{m-1}}] \ . \tag{2}$$

Definition 2. *ϕ mapping for vector $[x_0, x_1, \ldots, x_{m-1}]$ is defined as*

$$\phi[x_0, x_1, \ldots, x_{m-2}, x_{m-1}] := [x_{m-1}, x_0, x_1, \ldots, x_{m-2}] \ .$$

Definition 3. *"$a \sqsupseteq b$" is* true *if binary vector a includes binary vector b. More precisely,*

$$\begin{aligned}&[x_0, x_1, \ldots, x_{m-1}] \sqsupseteq [y_0, y_1, \ldots, y_{m-1}] \\ &:= \begin{cases} \mathsf{true} & \textit{if } x_j \vee \overline{y_j} = 1 \ (\textit{i.e., } x_j \geq y_j) \textit{ for } \forall j \ (0 \leq j < m) \\ \mathsf{false} & \textit{otherwise} \end{cases}\end{aligned}$$

Definition 4. *$w_{\mathrm{H}}(x)$: Hamming weight of x, i.e., number of non-zero elements in vector x.*

Definition 5. *$v_i := [1, e_{i,0}, e_{i,1}, \ldots, e_{i,m-2}]$, where $\{e_{i,j}\}_{j=0}^{m-2}$ is the binary representation of i $(0 \leq i < 2^{m-1})$ in the little endian fashion, i.e., $e_{i,j}$ satisfies the following equation.*

$$i = \sum_{j=0}^{m-2} e_{i,j} 2^j, \quad \textit{where } e_{i,j} \in \{0, 1\} \ . \tag{3}$$

For example, we have $v_1 = [1, 1, 0, 0, 0, 0, 0]$ and $v_{13} = [1, 1, 0, 1, 1, 0, 0]$ for $m = 7$.

3.2 Our Approach

This section describes the key idea of the cyclic window algorithm using an example.

We assume that k is already expanded to base-ϕ representation, for example,

$$k = 4 + 11\phi + 7\phi^2 + 2\phi^3 + 3\phi^4 + 1\phi^5 + 0\phi^6 \ ,$$

and $m = 7$. The expansion can be done using [4, Step 1 in Base-ϕ Scalar Multiplication Procedure], for example. Let n' be the size of the coefficients of the expansion.

We use the online precomputation table as follows.

$$\begin{aligned} Q_1 &= (v_1 \bullet [1, \phi, \ldots, \phi^{m-1}]^{\mathrm{T}})P = (1 + \phi)P \\ Q_2 &= (v_2 \bullet [1, \phi, \ldots, \phi^{m-1}]^{\mathrm{T}})P = (1 + \phi^2)P \\ Q_3 &= (v_3 \bullet [1, \phi, \ldots, \phi^{m-1}]^{\mathrm{T}})P = (1 + \phi + \phi^2)P \end{aligned}$$

Let L be the number of points in the table.

First, we denote the base-ϕ representation as binary representations (see Fig. 1). Second, we reduce 1s using signed-binary forms applying the condition $2^j(1 + \phi + \cdots + \phi^{m-1})P = \mathcal{O}$ [4][1] (see Fig. 2).

[1] Take care of choosing P. There may be a point that the equation does not hold, because $\mathbb{Z}[\phi]$ is not always an integral domain.

Coeff.		Bin. Rep.			
4	ϕ^0	0	1	0	0
11	ϕ^1	1	0	1	1
7	ϕ^2	0	1	1	1
2	ϕ^3	0	0	1	0
3	ϕ^4	0	0	1	1
1	ϕ^5	0	0	0	1
0	ϕ^6	0	0	0	0

Fig. 1. Binary Representation of the Coefficients in Base-ϕ Expansion

Coeff.		Bin. Rep.					Precomp.		
4	ϕ^0	0	$\boxed{1}$	$\boxed{-1}$	$\boxed{-1}$		$\boxed{1}$	$\boxed{1}$	$\boxed{1}$
11	ϕ^1	$\boxed{1}$	$\boxed{0}$	0	0		$\boxed{1}$	$\boxed{0}$	$\boxed{1}$
7	ϕ^2	0	$\boxed{1}$	0	0			$\boxed{1}$	$\boxed{1}$
2	ϕ^3	0	0	0	$\boxed{-1}$	$\cdots -\phi^3 P$	Q_1	Q_2	Q_3
3	ϕ^4	0	0	0	0				
1	ϕ^5	0	0	$\boxed{-1}$	0				
0	ϕ^6	0	0	$\boxed{-1}$	$\boxed{-1}$	$\cdots -\phi^6 Q_1$			

Fig. 2. How to Use the Precomputation Table

Since we can compute $\phi^i Q_l$ from precomputed point Q_l with a low level of complexity, we can use not only Q_l but also $\phi^i Q_l$ $(0 \le i \le m-1)$ referring to a precomputed point. Fig. 2 shows that $-\phi^6 Q_1$ can be used in the least significant bit. If we use the basic sliding window algorithm, the scalar multiplication for the least significant bits is computed by

$$-P - \phi^3 P - \phi^6 P \ ,$$

but we can "wrap" precomputed point Q_1 from ϕ^6 to ϕ^0, since $\phi^7 P = P$ holds. Thus, we can cyclically use the precomputed points as follows.

$$-\phi^3 P - \phi^6 Q_1$$

Following the above observations, we can compute kP using (4) with the precomputed points Q_1, Q_2, and Q_3 similar to the left-to-right binary algorithm for multiple bases [1, Algorithm 14.88 in p. 618]. It requires only four elliptic curve additions and three elliptic curve doublings after three elliptic curve additions and subtractions in the online precomputation stage.

$$kP = 2(2(2(\phi P) + Q_2) - \phi^5 Q_3) - \phi^3 P - \phi^6 Q_1 \tag{4}$$

3.3 Procedure

Using the notations defined above, we propose the cyclic window algorithm in Fig. 3[2]. The algorithm computes scalar multiplication, kP, but we assume that scalar k is already expanded to base-ϕ representation using [4, Steps 1 and 2 in Base-ϕ Scalar Multiplication Procedure] to hold

$$k = \sum_{i=0}^{m-1} c_i \phi^i \quad (c_i \ge 0) \ .$$

[2] It is trivial that we can omit the first elliptic curve doubling and addition in the main computation stage, but we do not describe this in the figure, since we describe the algorithm as simple as possible for easy understanding.

By using [8, the explanation for (14)], we assume $\log_2 c_i \le n \ (= n')$, where n is the size of p and n' is the maximum size of coefficient c_i. The algorithm inputs the structure of the online precomputation table, $\{u_0, u_1, \dots, u_L\}$. The structure is defined in Sect. 4.1.

Note that the algorithm is a generalization of Solinas' algorithm [5].

```
Input: $c_i$ $(0 \le c_i < 2^{n'})$, $P$, $m$, $\{u_0, u_1, \dots, u_L\}$ $(\subseteq \{v_0, v_1, \dots v_{2^{m-1}-1}\})$
Output: $\sum_{i=0}^{m-1} c_i \phi^i P$
Step 1: [Coefficient adjustment stage]
        Let $c_{i,j}$ be the $j$-th bit of $c_i$ and $d_j$ be a vector which is a collection of
        $c_{i,j}$s, i.e.,
        $c_i = \sum_{j=0}^{n'-1} c_{i,j} 2^j$, where $c_{i,j} \in \{0,1\}$
        and
        $d_j \leftarrow [c_{0,j}, c_{1,j}, \dots, c_{m-1,j}]$.
        If $w_{\mathrm{H}}(d_j) \ge (m+1)/2$ then
            $s_j \leftarrow -1$ and $d_j \leftarrow \overline{d_j}$,
        else
            $s_j \leftarrow 1$.
Step 2: [Online precomputation stage]
        Compute $Q_l \leftarrow u_l \bullet [P, \phi P, \phi^2 P, \dots, \phi^{m-1} P]^{\mathrm{T}}$, for $0 \le l \le L$.
Step 3: [Main computation stage]
        $R \leftarrow \mathcal{O}$
        for $j = n' - 1$ downto 0 do
            $R \leftarrow 2R$
            for $l \leftarrow L$ downto 0 do
                for $i \leftarrow 0$ to $m - 1$ do
                    if $d_j \sqsupseteq \phi^i u_l$ then
                        $d_j \leftarrow d_j \oplus \phi^i u_l$, $R \leftarrow R + s_j \phi^i Q_l$
                    end if
                end for $i$
            end for $l$
        end for $j$
        Output $R$.
```

Fig. 3. Cyclic Window Algorithm Procedure

4 Evaluation of Cyclic Window Algorithm

In this section, we evaluate the performance of the cyclic window algorithm applied to OEF for practically used parameters: the size of the maximum prime order subgroup is at least 160 bits and at most 256 bits.

4.1 Table Structure

Deciding on the structure of the online precomputation table is difficult. We propose the following heuristic strategy.

Let $T = \{u_0, u_1, \ldots, u_L\}$ be the subset of $\{v_0, v_1, \ldots, v_{2^{m-1}-1}\}$ that satisfies the following conditions:

- $u_0 = v_0$
- $\forall u_l [T \cap \{\phi^1(u_l), \ldots, \phi^{m-1}(u_l), \phi^0(\overline{u_l}), \phi^1(\overline{u_l}), \ldots, \phi^{m-1}(\overline{u_l})\} = \emptyset]$
- $w_{\mathrm{H}}(u_l) \le w_{\mathrm{H}}(u_{l+1})$
- $\sigma(l) < \sigma(l+1)$ if $w_{\mathrm{H}}(u_l) = w_{\mathrm{H}}(u_{l+1})$, where $\sigma(\cdot)$ is a permutation of the set $\{0, 1, \ldots, 2^{m-1}-1\}$ and satisfies $u_l = v_{\sigma(l)}$.
- $\sigma(L)$ is as small as possible.

4.2 Table Size

In this section, we evaluate the bound of the online precomputation table size, L. Since the extension degree is m and the table does not need to contain $\mathcal{O}$ and P, the table size is at most $2^m - 2$. However, we need only one point in

$$\left\{ uP \;\middle|\; u \in \{\phi^0(v_l), \phi^1(v_l), \ldots, \phi^{m-1}(v_l), \phi^0(\overline{v_l}), \phi^1(\overline{v_l}), \ldots, \phi^{m-1}(\overline{v_l})\} \right\} , \quad (5)$$

because the other points can be easily computed in the main computation stage. Since m is odd prime, the size of Set (5) is exactly $2m$. Thus, it is sufficient to choose

$$L = \frac{2^{m-1}-1}{m} - 1 \quad (6)$$

at most.

Table 1 gives the values derived from (6).

Table 1. Table Size Bound

Extension degree m	3	5	7	11	13	17	19
Bound of table size L	0	2	8	92	314	3854	13796

4.3 Small Extension Degree

In this section, we consider the preparation of a table which is at maximum. This case only needs at most 1 elliptic curve addition in the main loop in the main computation stage. However, of course, large m requires a higher degree of complexity of the online precomputation stage. Referring to Table 1, we study the case of a small m, that is, $m \leq 7$. In this case, we do not need a large table according to Table 1. The size of the table is at most 8. Because of the coupon collector's paradox, the probability that there exists a point in the table which is not referred in Step 3 in Fig. 3 is very low, if the size of the coefficients of the base-ϕ expansion n' is greater than $8 \log 8$ that is approximately equal to 17. On the other hand, n, which equals the size of p, is large because the prime order subgroup must be sufficiently large. Since n' is larger than $160/(m-1)$ which is approximately equal to 27, it overwhelms 17. Thus, maximizing L makes the cyclic window algorithm perform the fastest.

To summarize the above discussion, the cyclic window algorithm requires $\frac{2^{m-1}-1}{m} - 1$ elliptic curve additions for the online precomputation, $n'-1$ elliptic curve doublings and $(n'-1)(1-\frac{2}{2^m})$ elliptic curve additions for the main computation on average.

4.4 Large Extension Degree

In this section, we consider a large m value, say $m \geq 11$. As we consider the previous sections, to maximize the table size is not effective because the maximum table size is very large. It seems very difficult to analyze the precise complexity. Therefore, we computed the average elliptic curve additions in j-loop of Step 3 in Fig. 3 for all d_j, which is not applied to the "if" clause in Step 1 by a computer. Table 2 shows the computed results for m and L. Let $t_{m,L}$ be a value in Table 2. The cyclic window algorithm requires L elliptic curve additions for the online precomputation, $n'-1$ elliptic curve doublings and $(n'-1)t_{m,L}$ elliptic curve additions for the main computation on average.

Table 2 shows that the table construction described in Sect. 4.1 is not the best. There are some numbers greater than one that use less of the precomputation table. However, the tendency of the values in Table 2 shows that the greater the table size, the fewer matches.

4.5 Comparison with the Best Previous Result

To the best of the author's knowledge, Lim and Hwang's results [7,8] are the fastest previously reported. Their computational complexity is evaluated in [8, Sect. 5]. They tried to evaluate their algorithm for general cases, but precise numbers were not clear, since their evaluation contains parameters. However, they could derive precise numbers when m and n' were fixed. They showed numbers for $(m, n') = (7, 28)$, $(11, 16)$, and $(13, 14)$. We derive the optimal table

Table 2. Average Matches for Each Bit in the Coefficients of the Base-ϕ Expansion

	m			
L	11	13	17	19
0	4.15	5.03	6.83	7.74
1	3.09	3.70	4.94	5.57
2	2.67	3.20	4.30	4.85
3	2.45	2.93	3.89	4.37
4	2.31	2.78	3.69	4.16
5	2.17	2.64	3.51	3.95
6	2.08	2.53	3.36	3.79
7	2.02	2.45	3.23	3.63
8	1.97	2.39	3.17	3.57
9	1.96	2.33	3.12	3.51
10	1.93	2.26	3.04	3.42
11	1.90	2.20	2.97	3.33
12	1.90	2.16	2.94	3.28
13	1.87	2.11	2.88	3.22
14	1.84	2.07	2.83	3.16
15	1.83	2.06	2.80	3.11
16	1.82	2.03	2.79	3.09
17	1.80	2.05	2.78	3.09
18	1.79	2.04	2.75	3.07
19	1.78	2.03	2.72	3.05
20	1.77	2.01	2.69	3.01
21	1.76	1.98	2.66	2.98
22	1.75	1.97	2.63	2.95
23	1.74	1.96	2.60	2.92

size for the above parameters in order to draw a comparison to their results. However, we only compare the number of elliptic curve additions, since the cyclic window algorithm and Lim and Hwang algorithm require the same number of elliptic curve doublings.

When m equals 7, $L = 8$ is optimal choice according to Sect. 4.3[3]. Thus, the scalar multiplication requires

$$8 + (28 - 1)(1 - \frac{2}{2^7}) = 34.6$$

elliptic curve additions on average.

When m equals 11 or 13, we need to find the smallest

$$L + (n' - 1)t_{m,L} \tag{7}$$

using Table 2. We tried all combinations of (7). We found that $L = 6$ for $m = 11$ and $L = 7$ for $m = 13$ are the optimal choices[4]. These cases require

$$6 + (16 - 1)2.08 = 37.2 \quad \text{and} \quad 7 + (14 - 1)2.45 = 38.9$$

elliptic curve additions, respectively. We summarize the above numbers and compare with Lim and Hwang results in Table 3.

5 Implementation Examples

5.1 Parameters

We use the following parameters which are constructed [4] by the Weil conjecture, and the field parameters are shown in the upper columns in Table 4. Note that α is a root of $f(x)$ used by $\mathrm{GF}(p)[x]/(f(x))$, and we choose the base as a generator of the maximum prime order subgroup.

[3] This case requires 448 bytes for storing the online precomputation table, when using the parameter shown in Table 4.

[4] These cases require 264 and 364 bytes for storing the online precomputation table, respectively, when using the parameters shown in Table 4.

Table 3. Comparison with Lim and Hwang's Results

	Online precomp.		Main comp.		Total	
(m, n')	Ours	[8]	Ours	[8]	Ours	[8]
$(7, 28)$	8Ae	15Ae	34.6Ae+27De	41.2Ae+27De	42.6Ae+27De	56.2Ae+27De
$(11, 16)$	6Ae	13Ae	37.2Ae+15De	41.7Ae+15De	43.2Ae+15De	54.7Ae+15De
$(13, 14)$	7Ae	14Ae	38.9Ae+13De	44.1Ae+13De	45.9Ae+13De	58.1Ae+13De

"Ae" and "De" denote the computational cost of an elliptic curve addition and an elliptic curve doubling, respectively.

Word length: 16 bits $y^2 = x^3 - 3x - 172$

Maximum prime order in subgroups 3 7735447064 0784663733 8580162818 7749646114 9221530761 (exceeding 168 bits)

Base point $(10869\alpha^{12}+3898\alpha^{11}+15358\alpha^{10}+3782\alpha^{9}+4242\alpha^{8}+7589\alpha^{7}+5310\alpha^{6}+12599\alpha^{5}+10370\alpha^{4}+9316\alpha^{3}+8340\alpha^{2}+184\alpha+9573, 8924\alpha^{12}+9141\alpha^{11}+9472\alpha^{10}+8964\alpha^{9}+14633\alpha^{8}+4204\alpha^{7}+5379\alpha^{6}+13644\alpha^{5}+11470\alpha^{4}+15042\alpha^{3}+6518\alpha^{2}+15906\alpha+7391)$

Word length: 32 bits $y^2 = x^3 - 3x - 85$

Maximum prime order in subgroups 239 4696831448 0862150279 8948628438 5174133848 4034750169 (exceeding 174 bits)

Base point $(200472906\alpha^6 + 172723217\alpha^5 + 174386879\alpha^4 + 403718784\alpha^3 + 23043362\alpha^2 + 525400877\alpha + 17252111, 523133120\alpha^6 + 178522781\alpha^5 + 357710308\alpha^4 + 10611891\alpha^3 + 423928020\alpha^2 + 2135201\alpha + 535095305)$

5.2 Timings

Based on the above discussion, we implemented scalar multiplication in the elliptic curves. The timing is summarized in Table 4. In the table, "$\frac{1}{2}$" means that we adopted the parallel multiplication technique described in [11]. For example, we can compute two OEF multiplications in 604 cycles on a Pentium II. Table 4 also shows Lim and Hwang's results [7,8] as a reference. We refer to the detailed timings shown in [7]. However, we refer to the timings of the scalar multiplication shown in [8], since the timings of a scalar multiplication shown in [8] are faster than those in [7]. The results are scaled to 450 MHz for Pentium II and 500 MHz for 21164.

Our implementations on the Pentium II, 21164A, and 21264 use the Jacobian coordinate, the affine coordinate, and the coordinate proposed in [7, Sect. 2.2], respectively. We selected a 160-bit random integer as a scalar. Even if we select a number close to the order of subgroup generated by the base point as a scalar, the time for main computation stage hardly increases, but the time for converting a scalar to the base-ϕ representation will slightly increase. Note that $a = -3$ is always used in (1) for fast implementation purposes.

Table 4. Elliptic Curve Scalar Multiplication (cycles)

	Current study			[7]		
CPU	Pentium II	21164A	21264	Pentium II		21164
p	$2^{14}-3$	$2^{29}-3$	$2^{29}-3$	$2^{14}-3$	$2^{28}-57$	$2^{28}-57$
$f(x)$	$x^{13}-2$	x^7-2	x^7-2	$x^{13}-2$	x^7-2	x^7-2
Subgroup order	168	174	174	168	168	168
scalar mult (μs)	573	595	254	791	687	672
scalar mult (10^3)	258	298	127	356	309	336
EC add (A)	NA	3412	1544	6091	4628	4866
EC dbl (A)	NA	3830	1621	6863	5107	5543
EC add (P)	(M)3692	4152	1524	6171	4256	4696
EC dbl (P)	2528	3128	1164	4442	3086	3518
OEF inv	$\frac{1}{2}$4824	2120	1010	4200	3259	3292
OEF mult	$\frac{1}{2}$604	323	117	543	379	383
OEF sqr	$\frac{1}{2}$525	309	99	404	301	359
OEF ϕ	111	116	70			
OEF add	26	58	28	91	42	59
OEF sub	21	58	28			
GF(p) inv	1	266	219	19	457	376

- (A): affine, (P): projective
- (M): the addend is represented by affine coordinate
- NA: Not Available
- Pentium II (450 MHz), Other CPU (500 MHz)

6 Conclusion

This paper presented the cyclic window algorithm, a new scalar multiplication algorithm for elliptic curves defined over OEF. The algorithm first makes an online precomputation table and then computes a scalar multiplication using the precomputation table with the Frobenius map. The condition of the Frobenius map $\phi^m = 1$ allows us to use the precomputation table cyclically. This highly used Frobenius map makes scalar multiplication about 15% faster than the previously reported best results [7,8]. We also implemented our algorithm by software. A scalar multiplication can be computed in 573μs, 595μs, and 254μs on Pentium II (450 MHz), 21164A (500 MHz), and 21264 (500 MHz) computers, respectively.

Finally, how to decide the structure of the online precomputation table is left for future study.

References

1. Menezes, A.J., van Oorschot, P.C., Vanstone, S.A.: Handbook of applied cryptography. CRC Press (1997)
2. Bailey, D.V., Paar, C.: Optimal extension fields for fast arithmetic in public-key algorithms. In Krawczyk, H., ed.: Advances in Cryptology — CRYPTO'98. Volume 1462 of Lecture Notes in Computer Science. Springer-Verlag, Berlin, Heidelberg, New York (1998) 472–485
3. Koblitz, N.: CM-curves with good cryptographic properties. In Feigenbaum, J., ed.: Advances in Cryptology — CRYPTO'91. Volume 576 of Lecture Notes in Computer Science. Springer-Verlag, Berlin, Heidelberg, New York (1992) 279–287
4. Kobayashi, T., Morita, H., Kobayashi, K., Hoshino, F.: Fast elliptic curve algorithm combining frobenius map and table reference to adapt to higher characteristic. In Stern, J., ed.: Advances in Cryptology — EUROCRYPT'99. Volume 1592 of Lecture Notes in Computer Science. Springer-Verlag, Berlin, Heidelberg, New York (1999) 176–189 (A preliminary version was written in Japanese and presented at SCIS'99-W4-1.4).
5. Solinas, J.A.: An improved algorithm for arithmetic on a family of elliptic curves. In Kaliski Jr., B.S., ed.: Advances in Cryptology — CRYPTO'97. Volume 1294 of Lecture Notes in Computer Science. Springer-Verlag, Berlin, Heidelberg, New York (1997) 357–371
6. Lim, C.H., Lee, P.J.: More flexible exponentiation with precomputation. In Desmedt, Y.G., ed.: Advances in Cryptology — CRYPTO'94. Volume 839 of Lecture Notes in Computer Science. Springer-Verlag, Berlin, Heidelberg, New York (1994) 95–107
7. Lim, C.H., Hwang, H.S.: Fast implementation of elliptic curve arithmetic in GF(p^n). In Imai, H., Zheng, Y., eds.: Public Key Cryptography — Third International Workshop on Practice and Theory in Public Key Cryptosystems, PKC 2000. Volume 1751 of Lecture Notes in Computer Science. Springer-Verlag, Berlin, Heidelberg, New York (2000) 405–421
8. Lim, C.H., Hwang, H.S.: Speeding up elliptic scalar multiplication with precomputation. In Song, J.S., ed.: Information Security and Cryptology — ICISC'99. Volume 1787 of Lecture Notes in Computer Science. Springer-Verlag, Berlin, Heidelberg, New York (2000) 102–119
9. Tsuruoka, Y., Koyama, K.: Fast computation over elliptic curves $E(\mathbf{F}_{q^n})$ based on optimal addition sequences. IEICE Transactions Fundamentals of Electronics, Communications and Computer Sciences (Japan) **E84-A** (2001) 114–119
10. Cohen, H., Miyaji, A., Ono, T.: Efficient elliptic curve exponentiation using mixed coordinates. In Ohta, K., Pei, D., eds.: Advances in Cryptology — ASIACRYPT'98. Volume 1514 of Lecture Notes in Computer Science. Springer-Verlag, Berlin, Heidelberg, New York (1998) 51–65
11. Aoki, K., Hoshino, F., Kobayashi, T., Oguro, H.: Elliptic curve arithmetic using SIMD. In Davida, G., Frankel, Y., eds.: Information Security Conference — ISC'01. Lecture Notes in Computer Science. Springer-Verlag, Berlin, Heidelberg, New York (2001) *to appear.* (Preliminary version written in Japanese was appeared in SCIS2000-B05 and ISEC2000-161.).