Faster Elliptic Curve Arithmetic for Double-Base Chain by Reordering Sequences of Field Operations

Chitchanok Chuengsatiansup
Graduate School of Information Science and Technology
The University of Tokyo
Email: chitchanok@is.s.u-tokyo.ac.jp

***Abstract*—We have developed a new method for faster elliptic curve scalar multiplication represented in double-base chain format by cutting down redundancy using reordering sequences of field arithmetic operations. This method utilizes already-computed values obtained at some prior calculations to avoid unnecessary computations at some following calculations of a very time-consuming yet frequently executed scalar multiplication. We found that computing point doubling before point tripling reduces two computations of field squaring for curves defined over prime field, and consecutively point tripling or computing point tripling followed by point doubling reduces one computation of field squaring for curves defined over binary field. Experimental results showed achievements of 1.95% and 0.31% speed-up for curves defined over prime field and binary field respectively.**

I. Introduction

Speed of any elliptic curve cryptography protocols is related to speed of computing scalar multiplication which is known to be a very time-consuming operation. One of the crucial factors that affects speed of scalar multiplication relies on the underlying layer called field arithmetic operation.

Dimitrov, Imbert, and Mishra [1] showed that for curves defined over prime field in Jacobian coordinates, by reusing intermediate values, consecutive point triplings (w-TPL$^{\mathcal{J}}$) can save one computation of field squaring for each of the following point tripling; and consecutive point triplings followed by consecutive point doublings (w-TPL$^{\mathcal{J}}$/w'-DBL$^{\mathcal{J}}$) can save one computation of field squaring for the first point doubling that *follows* point tripling. There is another paper by Vuillaume, Okeya, and Takagi [2] mentioned about reordering *independent* operations in order to prevent differential attacks.

Our study revealed that by reordering *dependent* operations, i.e., computing consecutive point doublings *prior to* consecutive point triplings (w'-DBL$^{\mathcal{J}}$/w-TPL$^{\mathcal{J}}$), *more* intermediate values can be reused. That is, when the first point tripling that follows point doubling is computed, two computations of field squaring can be reduced compared to one in the previous work. We would like to emphasize that computing *point tripling after point doubling* is the main difference between previous and our work.

To precisely illustrate improvement achieved by applying our new method of reordering field arithmetic operation sequences, several experiments using randomly chosen 256-bit integers have been conducted. Our experimental results of computing point doubling *before* point tripling on curves defined over prime field in Jacobian coordinates showed a reduction of 1.55% assuming $[s] = 0.8[m]$ where $[s]$ and $[m]$ are cost of field squaring and field multiplication respectively. In case of using mixed addition, the reduction increased to 1.71%. If we assume $[s] = [m]$ in order to prevent simple side-channel analysis, we obtained reductions of 1.77% and 1.95% for general addition and mixed addition respectively.

In addition to curves defined over prime field, we also examined field arithmetic operation sequences on curves defined over binary field in Jacobian coordinates. Our study indicated that consecutive point triplings (w-TPL$^{\mathcal{J}}$) or consecutive point triplings before consecutive point doublings (w-TPL$^{\mathcal{J}}$/w'-DBL$^{\mathcal{J}}$) allows some intermediate values to be reused. In the former case, one computation of field squaring can be reduced for each of the following point tripling. In the latter case, one computation of field squaring can be reduced for point doubling that follows point tripling.

Our experimental results on curves defined over binary field in Jacobian coordinates showed reductions of 0.28% and 0.31% using general addition and mixed addition respectively. Even though this is a small step of improvement, it is still interesting from a theoretical view point that the idea of reusing intermediate value is applicable for curves defined over binary field. It is true for general cases that cost of field squaring is relatively small compared to cost of field multiplication. However, advancement in technology nowadays has made multiplication faster which affects the assumption that field squaring is relatively free compared to field multiplication.

II. Preliminaries

A. Elliptic Curve Cryptography

An elliptic curve E over a field K is defined by the equation

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \qquad (1)$$

where $a_1, a_2, a_3, a_4, a_6 \in K$.

If the characteristic of K is not equal to 2 or 3, the above equation can be simplified into $y^2 = x^3 + ax + b$ where $\Delta = 4a^3 + 27b^2 \neq 0$. If the characteristic of K is equal to 2, the ordinary form an elliptic curve is $y^2 + xy = x^3 + ax^2 + b$ where $\Delta = b \neq 0$. (see [3] for more details)

Scalar multiplication is one of the main operations in elliptic curve cryptography. This operation computes $[n]P = P + \cdots + P$ (n times) for a given point P and a scalar n.

B. Double-Base Number System

In double-base number system (DBNS) [4] [5], a number n can be represented as a summation of mixed power of two co-prime integers as follows:

$$n = \sum_i d_i p^{a_i} q^{b_i} \tag{2}$$

where $d_i \in \{-1, 1\}$, $\gcd(p, q) = 1$, and $a_i, b_i \geq 0$.

An example of DBNS using mixed power of 2 and 3 is $314159 = 2^7 3^7 + 2^{10} 3^3 + 2^1 3^8 + 2^2 3^1 + 2^1$. One advantage of this representation over a single-base format is that fewer number of terms in the expansion can be achieved. This is also the reason why DBNS has recently gained much attention in a research area of elliptic curve cryptography.

If extra restrictions over the exponent requiring $a_1 \geq a_2 \geq \cdots \geq a_\ell$ and $b_1 \geq b_2 \geq \cdots \geq b_\ell$ are also considered, this special type of DBNS is called *double-base chain* which was introduced in [5]. An example of double-base chain representation is $314159 = 2^4 3^9 - 3^6 - 3^3 - 3^2 - 1$. Double-base chain representation is more desirable since it allows better efficiency to compute $[n]P$. For example, $[314159]P$ can be computed as $[3^2]([3^1]([3^3]([2^4 3^3]P - P) - P) - P) - P$.

To use double-base chain representation, the original integer has to be converted into double-base chain format. Fig. 1 shows an example of algorithm converting an integer n into (2,3) double-base chain representation.

```
Require: $n > 0$ and $a_{max}, b_{max}$[1] $> 0$
Ensure: sequence $(d_i, a_i, b_i)$ such that $n = \sum_i d_i 2^{a_i} 3^{b_i}$ with
    $a_1 \geq a_2 \geq \cdots \geq a_\ell$ and $b_1 \geq b_2 \geq \cdots \geq b_\ell$
 1: $d \leftarrow 1$
 2: while $n > 0$ do
 3:     $r \leftarrow 2^a 3^b$, best approximation of $n$ with $0 \leq a \leq a_{max}$
        and $0 \leq b \leq b_{max}$
 4:     $a_{max} \leftarrow a$, $b_{max} \leftarrow b$
 5:     print $(d, a, b)$
 6:     if $r > n$ then
 7:         $d \leftarrow -1$
 8:     end if
 9:     $n \leftarrow |n - r|$
10: end while
```

[1] a_{max} and b_{max} are the maximum values allowed for binary and ternary exponents respectively

Fig. 1. Algorithm converting an integer n into $(2, 3)$ double-base *chain*

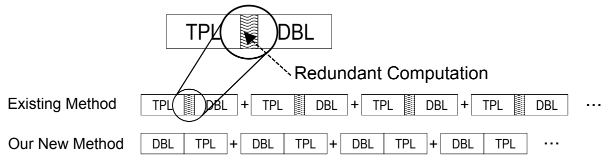
C. Consecutive Field Arithmetic Operations

Cohen, Miyaji, and Ono [6] explained that when point doubling is computed consecutively , some intermediate values from prior point doubling can be reused so that each of the following point doubling can be saved up to two computations of field squaring. Dimitrov, Imbert, and Mishra [1] showed that when point tripling is computed consecutively or when point doubling is computed after point tripling, some intermediate values from previous computations can reduce one computation of field squaring for each of the following operation.

III. Our Method

A. Elliptic Curves Defined over Prime Field

As mentioned in the previous section that in prime field of Jacobian coordinates, [1] described that computing point tripling *before* point doubling can reduce one field squaring. Our study revealed that by slightly reordering, namely, computing point tripling *after* point doubling, two field squarings can be reduced. Fig. 2 illustrates our new method of reordering field arithmetic sequences.



TPL
DBL
Redundant Computation
Existing Method
TPL DBL + TPL DBL + TPL DBL + TPL DBL ···
Our New Method
DBL TPL + DBL TPL + DBL TPL + DBL TPL ···

Fig. 2. New method of reordering field arithmetic operations to reduce redundant computations for curves defined over prime field

We carefully examined point doubling and point tripling operations for curves defined over prime field in Jacobian coordinates and analyzed values needed to be computed in each step. We shall refer to those values as *intermediate values*.

For example, to compute $2X_1Y_1^2$, we first need to compute Y_1^2. Then, we multiply X_1 by Y_1^2 to obtain $X_1Y_1^2$. Finally, we add $X_1Y_1^2$ to itself and get $2X_1Y_1^2$. In this example, Y_1^2 and $X_1Y_1^2$ are intermediate values. The value Y_1^2 is obtained by squaring Y_1 while the value $X_1Y_1^2$ is obtained by multiplying X_1 and Y_1^2 together.

Let $P_1 = (X_1, Y_1, Z_1)$ be a point in Jacobian coordinates on the elliptic curve E, and let $P_2 = [2]P_1 = (X_2, Y_2, Z_2)$ be the result of doubling point P_1. Table I summarizes intermediate values occurred during each step of computing point doubling. We categorized them into values obtained by field multiplication and field squaring.

TABLE I
INTERMEDIATE VALUES OCCURRED DURING POINT DOUBLING ON CURVES DEFINED OVER PRIME FIELD IN JACOBIAN COORDINATES

Operation	Intermediate Values	
	Multiplication	Squaring
$\alpha_2 = 3X_1^2 + aZ_1^4$	aZ_1^4	X_1^2, Z_1^2, Z_1^4
$\beta_2 = 4X_1Y_1^2$	$X_1Y_1^2$	Y_1^2
$X_2 = \alpha_2^2 - 2\beta_2$	$-$	α_2^2
$Y_2 = \alpha_2(\beta_2 - X_2) - 8Y_1^4$	$\alpha_2(\beta_2 - X_2)$	Y_1^4
$Z_2 = 2Y_1Z_1$	Y_1Z_1	$-$

According to Table I, to compute α_2, it is necessary that aZ_1^4 is computed. Similarly, Y_1^4 needs to be computed in order to compute Y_2. These two intermediate values, namely, aZ_1^4 and Y_1^4, will appear again when point tripling is computed afterwards.

Let $P_3 = [3]P_2 = (X_3, Y_3, Z_3)$ be the result of tripling point P_2. Table II summarizes intermediate values occurred during each step of computing point tripling. We also categorized them into values obtained by field multiplication and field squaring.

TABLE II
INTERMEDIATE VALUES OCCURRED DURING POINT TRIPLING ON CURVES DEFINED OVER PRIME FIELD IN JACOBIAN COORDINATES

Operation	Intermediate Values	
	Multiplication	Squaring
$\theta_3 = 3X_2^2 + aZ_2^4$	aZ_2^4	X_2^2, Z_2^2, Z_2^4
$\omega_3 = 12X_2Y_2^2 - \theta_3^2$	$X_2Y_2^2$	Y_2^2, θ_3^2
$\alpha_3 = \theta_3\omega_3$	$\theta_3\omega_3$	$-$
$\beta_3 = 8Y_2^4$	$-$	Y_2^4
$X_3 = 8Y_2^2(\beta_3 - \alpha_3) + X_2\omega_3^2$	*	ω_3^2
$Y_3 = Y_2[4(\alpha_3 - \beta_3)(2\beta_3 - \alpha_3) - \omega_3^3]$	**	$-$
$Z_3 = 2Z_2\omega_3$	$Z_2\omega_3$	$-$

* $Y_2^2(\beta_3 - \alpha_3), X_2\omega_3^2$
** $(\alpha_3 - \beta_3)(2\beta_3 - \alpha_3), \omega_3^3, Y_2[4(\alpha_3 - \beta_3)(2\beta_3 - \alpha_3) - \omega_3^3]$

In this work, we found that by substituting $Z_2 = 2Y_1Z_1$ from point doubling formula and rewriting θ_3 as $3X_2^2 + 16aZ_1^4Y_1^4$, the intermediate values aZ_1^4 and Y_1^4 can be reused to reduce two computations of field squaring. In other words, we can compute θ_3 by calculating only one squaring for X_2^2 and one multiplication for $(aZ_1^4)(Y_1^4)$ while directly computing from point tripling formula requires three squarings for X_2^2, Z_2^2, Z_2^4 and one multiplication for aZ_2^4. Table III shows how we reused intermediate values to reduce two computations of field squaring when point tripling is computed after point doubling.

TABLE III
IDEA OF REDUCING TWO SQUARINGS WHEN COMPUTING POINT TRIPLING AFTER POINT DOUBLING ON CURVES DEFINED OVER PRIME FIELD

Operation	Intermediate Values	
	Multiplication	Squaring
$\theta_3 = 3X_2^2 + aZ_2^4$	aZ_2^4	$\boldsymbol{X_2^2}, \boldsymbol{Z_2^2}, Z_2^4$
$\theta_3 = 3X_2^2 + a[2Y_1Z_1]^4$ $\theta_3 = 3X_2^2 + 2^4 \cdot aZ_1^4Y_1^4$	$(aZ_1^4)(Y_1^4)$	X_2^2

Note that in [1], they considered the cost of point tripling to be $10[m]+6[s]$. However, there exists *revised* tripling formula which uses $9[m] + 7[s]$. We derived our method from this revised tripling formula. Hence, we compared our results to the previous idea [1] of computing point tripling *before* point doubling (w-TPL$^{\mathcal{J}}$/w'-DBL$^{\mathcal{J}}$) combining with the revised tripling formula. Table IV shows the cost comparison between previous work [1] [7], namely, computing point tripling *before* point doubling (w-TPL$^{\mathcal{J}}$/w'-DBL$^{\mathcal{J}}$) with revised tripling, and this work, namely, computing point tripling *after* point doubling (w'-DBL$^{\mathcal{J}}$/w-TPL$^{\mathcal{J}}$).

TABLE IV
COST COMPARISON BETWEEN DIFFERENT ORDERS OF POINT DOUBLING AND POINT TRIPLING ON CURVES DEFINED OVER PRIME FIELD

Operation	Cost
Previous results [1]	$(9w+4w')[m]+(6w+4w'+\mathbf{2})[s]$
Our results [2]	$(9w+4w')[m]+(6w+4w'+\mathbf{1})[s]$

[1] w-TPL$^{\mathcal{J}}$/w'-DBL$^{\mathcal{J}}$ [1] [7]
[2] w'-DBL$^{\mathcal{J}}$/w-TPL$^{\mathcal{J}}$

Example: The following explains how our method works compared to previous method [1] [7]. In this example, we would like to compute $[1739]P$ from a given double-base chain $1739 = 2^63^3 + 2^23^1 - 2^03^0 = [2^23^1]([2^43^2]P + P) - P$.

In previous work, $[1739]P$ would be computed as follows:

1) $[3]P$ is computed using $9[m] + 7[s]$, then $[3]([3]P)$ is computed using $9[m] + 6[s]$.
Total cost of $18[m] + 13[s]$ is required at step 1.
2) $[2]([3^2]P)$ is computed using $4[m] + 5[s]$, then $[2]([2^13^2]P)$, $[2]([2^23^2]P)$, $[2]([2^33^2]P)$ is computed using $4[m] + 4[s]$ at each step.
Total cost of $16[m] + 17[s]$ is required at step 2.
3) $[2^43^2]P + P$ is computed using $8[m] + 3[s]$.
(The rest of the steps are similar to step 1-3. Thus, we leave out the explanation.)

Summation of field multiplication and field squaring needed from step 1 to 3 is $42[m] + 33[s]$. Including the rest of the steps, total cost of $67[m] + 52[s]$ is required.

On the other hand, in our method, $[1739]P$ would be computed as follows:

1) $[2]P$ is computed using $4[m] + 6[s]$, then $[2]([2]P)$, $[2]([2^2]P)$, $[2]([2^3]P)$ is computed using $4[m] + 4[s]$ at each step.
Total cost of $16[m] + 18[s]$ is required at step 1.
2) $[3]([2^4]P)$ is computed using $9[m] + 5[s]$, then $[3]([2^43^1]P)$ is computed using $9[m] + 6[s]$.
Total cost of $18[m] + 11[s]$ is required at step 2.
3) $[2^43^2]P + P$ is computed using $8[m] + 3[s]$.
(The rest of the steps are similar to step 1-3. Thus, we leave out the explanation.)

Summation of field multiplication and field squaring needed from step 1 to 3 is $42[m] + 32[s]$. Including the rest of the steps, total cost of $67[m] + 50[s]$ is required.

We can see that step 1 to 3 computes one term in double-base chain ($[2^43^2]P+P$ in this case). In these three steps, our method reduces one squaring; that is, $32[s]$ compared to $33[s]$. If the entire chain is computed, two squarings are reduced; that is, $50[s]$ compared to $52[s]$. This is because there are three terms in this expansion.

One thing that we would like to mention is for Barreto-Naehrig curve, the method of reusing intermediate values cannot be applied. Neither the previous work of computing point tripling before point doubling nor our work of computing point tripling after point doubling can profit from this technique.

B. Elliptic Curves Defined over Binary Field

We also examined point doubling and point tripling operations for curves defined over binary field in Jacobian coordinates and analyzed values needed to be computed in each step. We found that consecutive point triplings (w-TPL$^{\mathcal{J}}$) can reduce one computation of field squaring for each of the following point tripling, and consecutive point triplings before consecutive point doublings (w-TPL$^{\mathcal{J}}$/w'-DBL$^{\mathcal{J}}$) can reduce one computation of field squaring for the first point doubling that follows point tripling.

Let $P_1 = (X_1, Y_1, Z_1)$ be a point in Jacobian coordinates on the elliptic curve E defined over binary field, let $P_2 = [2]P_1 = (X_2, Y_2, Z_2)$ be the result of doubling point P_1, and let $P_3 = [3]P_1 = (X_3, Y_3, Z_3)$ be the result of tripling point P_1. Point P_3 is computed by adding the doubling of point P_1 to the original point P_1, i.e., $P_3 = [2]P_1 + P_1$. Table V summarizes intermediate values occurred during each step of computing point tripling on curves defined over binary field in Jacobian coordinates. We categorized them into values obtained by field multiplication and field squaring.

TABLE V
INTERMEDIATE VALUES OCCURRED DURING POINT TRIPLING ON CURVES DEFINED OVER BINARY FIELD IN JACOBIAN COORDINATES

Operation	Intermediate Values	
	Multiplication	Squaring
$X_2{=}X_1^4 + a_6Z_1^8$	$a_6Z_1^8$	$X_1^2, X_1^4, Z_1^2, Z_1^4, Z_1^8$
$Y_2 = X_1^4Z_2 + (X_1^2 + Y_1Z_1 + Z_2)X_2$	$*$	$-$
$Z_2 = X_1Z_1^2$	$X_1Z_1^2$	$-$
$\alpha_3 = a_2\gamma_3^2 + \omega_3(\omega_3 + \gamma_3) + \theta_3^3$	$**$	$\boldsymbol{\gamma_3^2}, \theta_3^2$
$\beta_3 = (\omega_3 + \gamma_3)\alpha_3 + \theta_3^2(\omega_3X_2 + \theta_3Y_2)$	$***$	$-$
$\gamma_3 = \theta_3Z_2$	θ_3Z_2	$-$
$\theta_3 = X_1^3Z_1^2 + X_2$	$(X_1Z_1^2)(X_1^2)$	$-$
$\omega_3 = Y_1X_1^3Z_1^3 + Y_2$	$****$	$-$
$X_3{=}\alpha_3Z_1^6$	$-$	$-$
$Y_3 = \beta_3Z_1^9$	$-$	$-$
$Z_3 = \gamma_3Z_1^3$	$-$	$-$

* $X_1^4Z_2, Y_1Z_1, (X_1^2 + Y_1Z_1 + Z_2)X_2$
** $a_2\gamma_3^2, \omega_3(\omega_3 + \gamma_3), \theta_3^3$
*** $(\omega_3 + \gamma_3)\alpha_3, \omega_3X_2, \theta_3Y_2, \theta_3^2(\omega_3X_2 + \theta_3Y_2)$
**** $(X_1^3Z_1^2)(Y_1Z_1)$

We can see from Table V that squaring of γ_3 is calculated when computing α_3. By using that fact that $[3]P_1 = (X_3, Y_3, Z_3) = (\alpha_3Z_1^6, \beta_3Z_1^9, \gamma_3Z_1^3) = (\alpha_3, \beta_3, \gamma_3)$, calculating γ_3^2 is equivalent to calculating Z_3^2. We would like to emphasize that the value Z_3^2 can be reused when point tripling or point doubling is computed afterwards.

Let $P_4 = [2]P_3 = (X_4, Y_4, Z_4)$ be the result of doubling point P_3, namely, computing point doubling after point tripling. Table VI summarizes intermediate values occur during point doubling on curved defined over binary field in Jacobian coordinates. Since the reusing intermediate values concept in point tripling is similar to point doubling, we leave out the explanation for the case of point tripling.

TABLE VI
INTERMEDIATE VALUES OCCURRED DURING POINT DOUBLING ON CURVES DEFINED OVER BINARY FIELD IN JACOBIAN COORDINATES

Operation	Intermediate Values	
	Multiplication	Squaring
$X_4{=}X_3^4 + a_6Z_3^8$	$a_6Z_3^8$	$X_3^2, X_3^4, Z_3^2, Z_3^4, Z_3^8$
$Y_4 = X_3^4Z_4 + (X_3^2 + Y_3Z_3 + Z_4)X_4$	$*$	$-$
$Z_4 = X_3Z_3^2$	$X_3Z_3^2$	$-$

* $X_3^4Z_4, Y_3Z_3, (X_3^2 + Y_3Z_3 + Z_4)X_4$

According to Table VI, Z_3^8 must be computed in order to obtain the value X_4. This implicitly implies that the value Z_3^2 needs to be calculated. We found that the value Z_3^2 does not need to be computed at this step because it has already been computed when computing γ_3^2. Table VII shows how we reused intermediate values to reduce one computation of field squaring.

TABLE VII
IDEA OF REDUCING ONE SQUARING WHEN COMPUTING POINT DOUBLING AFTER POINT TRIPLING ON CURVES DEFINED OVER BINARY FIELD

Operation	Intermediate Values	
	Multiplication	Squaring
$X_4{=}X_3^4 + a_6Z_3^8$	$a_6Z_3^8$	$X_3^2, X_3^4, \boldsymbol{Z_3^2}, Z_3^4, Z_3^8$
$\alpha_3 = a_2\gamma_3^2 + \omega_3(\omega_3 + \gamma_3) + \theta_3^3$	$*$	$\boldsymbol{\gamma_3^2}, \theta_3^2$
$X_4{=}X_3^4 + a_6Z_3^8$	$a_6Z_3^8$	$X_3^2, X_3^4, Z_3^4, Z_3^8$ $(\gamma_3^2 = Z_3^2)$

* $a_2\gamma_3^2, \omega_3(\omega_3 + \gamma_3), \theta_3^3$

Table VIII summarizes the cost of consecutive operations, namely, consecutive point triplings and consecutive point triplings followed by consecutive point doublings for curves defined over binary field in Jacobian coordinates after applying the method of reusing intermediate values. For references, cost of single point doubling and single point tripling are also displayed.

TABLE VIII
COST FOR CONSECUTIVE OPERATIONS ON CURVES DEFINED OVER BINARY FIELD WHEN APPLYING THE METHOD OF REUSING INTERMEDIATE VALUES

Operation	Cost
DBL$^{\mathcal{J}}$	$5[m]{+}5[s]$ [1]
TPL$^{\mathcal{J}}$	$15[m]{+}7[s]$ [1]
w-TPL$^{\mathcal{J}}$	$15w[m]{+}(6w{+}1)[s]$
w-TPL$^{\mathcal{J}}$/w'-DBL$^{\mathcal{J}}$	$(15w{+}5w')[m]{+}(6w{+}5w')[s]$

[1] Previous results [1]

IV. EXPERIMENTAL RESULTS

To compare the efficiency of our proposed method with previous results, several experiments have been conducted. In each experiment, we used $10,000$ randomly chosen 256-bit integers. Those integers were then converted into double-base *chain* representation of mixed power of two and three by the algorithm described in Fig. 1 of Section II-B.

Since we were finding the double-base *chain* representation, the restrictions $a_1 \geq a_2 \geq \cdots \geq a_\ell$ and $b_1 \geq b_2 \geq \cdots \geq b_\ell$ must also satisfy. Each experiment, we set a_{max} and b_{max} to 150 and 100 respectively. The a_{max} and b_{max} values do have effect on the efficiency because setting them too small or too large resulted in longer expansion. Longer expansion also means longer time required when computing scalar multiplication. According to [1], the best a_{max} and b_{max} for 160-bit integers are 95 and 41 respectively. We adjusted those numbers to fit 256-bit integers in our experiments.

For curves defined over prime field, it is generally assumed that $[s] = 0.8[m]$ where $[s]$ and $[m]$ are the cost of field squaring and field multiplication respectively. However, if we use side-channel atomic blocks introduced in [8] in order to prevent simple side-channel analysis, we have to consider $[s] = [m]$. We conducted the experiment concerning both cases of $[s] = [m]$ and $[s] = 0.8[m]$.

Table IX shows cost comparison of curves defined over prime field between previous results, namely, consecutive point triplings followed by consecutive point doublings , and our results, namely, consecutive point doublings followed by consecutive point triplings. ADD$^{\mathcal{J}}$ denotes general addition in Jacobian coordinates while ADD$^{\mathcal{J}+\mathcal{A}}$ denotes mixed addition in Jacobian-affine coordinates.

TABLE IX
COST COMPARISON BETWEEN PREVIOUS AND OUR RESULTS FOR CURVES DEFINED OVER PRIME FIELD IN JACOBIAN COORDINATES (IMPROVEMENTS COMPARED TO PREVIOUS RESULTS SHOWN IN PARENTHESES)

		Cost	$[s] = [m]$	$[s] = 0.8[m]$
ADD$^{\mathcal{J}}$	Previous results [1]	$1932.55[m]$ $+1336.00[s]$	3268.55 $(-)$	3001.35 $(-)$
	Our results [2]	$1932.55[m]$ $+1277.77[s]$	3210.32 (1.77%)	2954.77 (1.55%)
ADD$^{\mathcal{J}+\mathcal{A}}$	Previous results [1]	$1699.63[m]$ $+1277.77[s]$	2977.40 $(-)$	2721.85 $(-)$
	Our results [2]	$1699.63[m]$ $+1219.54[s]$	2919.17 (1.95%)	2675.26 (1.71%)

[1] w-TPL$^{\mathcal{J}}$/w'-DBL$^{\mathcal{J}}$ [1]
[2] w'-DBL$^{\mathcal{J}}$/w-TPL$^{\mathcal{J}}$

Experimental results using general addition in Jacobian coordinates showed reductions of 1.77% and 1.55% assuming $[s] = [m]$ and $[s] = 0.8[m]$ respectively. If mixed addition was used, the reduction increased to 1.95% and 1.71%.

For curves defined over binary field, in most cases it is assumed that cost of field squaring is very small and negligible compared to cost of field multiplication. However, with the continuous improvement of multiplication, this assumption becomes invalid in some situations. That is, cost of field squaring can not simply be neglected. In our experiment, we assumed the cost of field squaring and field multiplication to approximately be $[s] = 0.1[m]$ according to [9].

Table X shows the cost comparison for computing consecutive point triplings followed by consecutive point doublings (w-TPL$^{\mathcal{J}}$/w'-DBL$^{\mathcal{J}}$) on curves defined over binary field in Jacobian coordinates between applying and not applying our technique of reusing intermediate values.

TABLE X
COST COMPARISON BETWEEN APPLYING AND NOT APPLYING OUR TECHNIQUE FOR COMPUTING w-TPL$^{\mathcal{J}}$/w'-DBL$^{\mathcal{J}}$ ON CURVES DEFINED OVER BINARY FIELD (IMPROVEMENTS SHOWN IN PARENTHESES)

	Applying our technique	Cost	$[s] = 0.1[m]$
ADD$^{\mathcal{J}}$	No	$2785.18[m]$+$1365.84[s]$	$(-)$
	Yes	$2785.18[m]$+$1282.86[s]$	(0.28%)
ADD$^{\mathcal{J}+\mathcal{A}}$	No	$2494.60[m]$+$1365.84[s]$	$(-)$
	Yes	$2494.60[m]$+$1282.86[s]$	(0.31%)

Experimental results showed a reduction of 0.28% for general addition in Jacobian coordinates. For mixed addition, the reduction increased to 0.31%.

V. CONCLUSION

Same integer n with the same double-base chain represented but computed in different orders leads to different amounts of time required. We raised a discussion on sequences of field arithmetic operation for computing elliptic curve scalar multiplication in double-base chain representation. For curves defined over prime field, a traditional method to compute scalar multiplication in (2,3) double-base chain representation is to compute point tripling then point doubling. It is trivial to see that by switching the sequences, that is, computing point doubling before point tripling, also yields the same result. Nevertheless, its consequence of less computation time required has not yet been revealed.

Scalar multiplication enhancement has long been studied. However, modifications on upper layers are more application specific and can be utilized only in narrow scopes. Unlike our method, we aimed to improve at fundamental layers in order to be applicable to more extensive range of areas. Because field arithmetic operation is a building block for many computations including scalar multiplication in elliptic curve cryptography, speeding up at field arithmetic level can further improve many applications in higher levels.

Our ideas of solving problem in another direction, i.e., *reordering* field arithmetic operation sequences and reusing *intermediate* values, are expected to be inspirations to others within the same and different areas of research on how to look at and understand problems.

REFERENCES

[1] V. S. Dimitrov, L. Imbert, and P. K. Mishra, "The double-base number system and its application to elliptic curve cryptography," *Mathematics of Computation*, vol. 77, no. 262, pp. 1075–1104, 2008.

[2] C. Vuillaume, K. Okeya, and T. Takagi, "Short-memory scalar multiplication for koblitz curves," *IEEE Transactions on Computers*, vol. 57, no. 4, pp. 481–489, April 2008.

[3] *Elliptic Curves: Number Theory and Cryptography*. Chapman and Hall/CRC, 2008.

[4] V. Dimitrov, G. Jullien, and W. Miller, "Theory and applications of the double-base number system," *IEEE Transactions on Computers*, vol. 48, no. 10, pp. 1098 –1106, oct 1999.

[5] C. Doche and L. Imbert, "Extended double-base number system with applications to elliptic curve cryptography," in *Proceedings of the 7th international conference on Cryptology in India*, ser. INDOCRYPT'06. Berlin, Heidelberg: Springer-Verlag, 2006, pp. 335–348.

[6] H. Cohen, A. Miyaji, and T. Ono, "Efficient elliptic curve exponentiation using mixed coordinates," in *Proceedings of the International Conference on the Theory and Applications of Cryptology and Information Security: Advances in Cryptology*, ser. ASIACRYPT '98. London, UK, UK: Springer-Verlag, 1998, pp. 51–65.

[7] P. Longa and A. Miri, "Fast and flexible elliptic curve point arithmetic over prime fields," *IEEE Transactions on Computers*, vol. 57, no. 3, pp. 289–302, March 2008.

[8] B. Chevallier-Mames, M. Ciet, and M. Joye, "Low-cost solutions for preventing simple side-channel analysis: Side-channel atomicity," *IEEE Transactions on Computers*, vol. 53, no. 6, pp. 760–768, June 2004.

[9] D. Hankerson, J. L. Hernandez, and A. Menezes, "Software implementation of elliptic curve cryptography over binary fields," in *Proceedings of the Second International Workshop on Cryptographic Hardware and Embedded Systems*, ser. CHES '00. London, UK, UK: Springer-Verlag, 2000, pp. 1–24.