Evaluating Optimized Computation in Double-Base Chain for Efficient Elliptic Curve Cryptography

Chitchanok Chuengsatiansup, Hiroshi Imai, and Vorapong Suppakitpaisarn

Graduate School of Information Science and Technology
The University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo, JAPAN 113-0033
{chitchanok,imai,mr_t_dtone}@is.s.u-tokyo.ac.jp

We have developed a method to accelerate scalar multiplication in elliptic curve cryptography for number represented in double-base chain by removing redundant computation and reordering sequences of field arithmetic operation. This method reuses intermediate values obtained from previous steps to remove unnecessary calculations in following steps. Our study revealed that computing point doubling prior to point tripling can reduce two computations of field squaring for curves defined over prime field. On the other hand, computing point tripling prior to point doubling can reduce one computation of field squaring for curves defined over binary field. Experimental results showed a speed-up of 1.95 % if solely our method of reusing intermediate value was used. Moreover, our method can be integrated with a technique previously proposed by Longa and Miri on converting costly field multiplication into cheaper field squaring.

1 Introduction

Scalar multiplication is one of the most important operations in elliptic curve cryptography. Speed of scalar multiplication partially relies on the underlying layer called field arithmetic operation.

Dimitrov, Imbert, and Mishra [6] introduced a technique of reusing intermediate values to reduce some computations when operations on curves defined over prime field in Jacobian coordinates are executed consecutively. That is, when point tripling is computed consecutively (w-TPL$^{\mathcal{J}}$), except for the first point tripling, one field squaring can be saved for each of the following point tripling. Similarly, when computing consecutive point triplings followed by consecutive point doublings (w-TPL$^{\mathcal{J}}$/w'-DBL$^{\mathcal{J}}$), one field squaring can be saved for the first point doubling that follows point tripling.

According to our study, the sequences of field operation are crucial to how intermediate values can be reused. We found that by altering the order, i.e., computing consecutive point doublings *before* consecutive point triplings (w'-DBL$^{\mathcal{J}}$/w-TPL$^{\mathcal{J}}$), *more* intermediate values can be reused. In other words, two computations of field squaring can be reduced (compared to one in the previous work) if computing point tripling after point doubling. In fact, computing *point tripling after point doubling* is a main difference between previous and our work.

We conducted several experiments using $10,000$ randomly chosen 256-bit integers to evaluate improvement achieved by applying our new method of reordering sequences of field arithmetic operation. Our experimental results showed that for curves defined over prime field in Jacobian coordinates, 1.55% of a computational cost was reduced if computing point doubling

before point tripling under an assumption that $[s] = 0.8[m]$ where $[s]$ and $[m]$ are the 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, the reduction changed to 1.77% and 1.95% for general addition and mixed addition respectively.

Our technique of reusing intermediate value can also be applied together with the concept of converting expensive field multiplication into cheaper field squaring or S-M trade-off proposed by Longa and Miri [8]. That is, speed-up from two techniques, namely, S-M trade-off and reusing intermediate value, is obtainable at the same time.

We also examined the possibility to speed-up computation on curves defined over binary field in Jacobian coordinates by reusing intermediate values. Results of our study showed that one field squaring can be reduced if point tripling is computed consecutively (w-TPL$^{\mathcal{J}}$) and when computing consecutive point triplings before consecutive point doublings (w-TPL$^{\mathcal{J}}$/w'-DBL$^{\mathcal{J}}$).

Experimental results on curves defined over binary field in Jacobian coordinates showed reductions of 0.28% and 0.31% if general addition and mixed addition were used respectively. Although these figures are rather small, they show an interesting fact that reusing intermediate value is possible for curves defined over binary field.

2 Preliminaries

2.1 Elliptic Curve Cryptography

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

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \tag{1}$$

where $a_1, a_2, a_3, a_4, a_6 \in K$, and $\Delta \neq 0$, where Δ is the discriminant of E.

Equation (1) can be rewritten into simpler forms depending on the characteristic of the underlying fields. If binary field is used, equation (1) can be simplified to $y^2+xy = x^3+ax^2+b$ where $\Delta = b \neq 0$. If prime field is used, equation (1) can be further simplified to $y^2 = x^3+ax+b$ where $\Delta = 4a^3 + 27b^2 \neq 0$. (see [1] 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 on the elliptic curve E and a secret scalar n.

2.2 Double-Base Number System

In double-base number system (DBNS) [5], a number n is 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 $a_i, b_i \geq 0$.

For example, 314159 can be represented using DBNS of mixed power of two and three as $2^73^7 + 2^{10}3^3 + 2^13^8 + 2^23^1 + 2^1$. DBNS allows shorter expansions (fewer number of terms) compared to binary representation. This is one of the reasons that many researchers in elliptic curve cryptography are much interested in DBNS.

Double-base chain representation, introduced in [4], is a special type of DBNS where the extra restrictions over the exponent, that is, $a_1 \geq a_2 \geq \cdots \geq a_\ell$ and $b_1 \geq b_2 \geq \cdots \geq b_\ell$, are also considered. One example of double-base chain representation is $314159 = 2^43^9-3^6-3^3-3^2-1$. Double-base chain is preferable to (regular) double-base number system because it allows an efficient method to compute scalar multiplication. That is, $[314159]P$ can be computed as $[3^2]([3^1]([3^3]([2^43^3]P - P) - P) - P) - P$.

To use double-base chain representation, the original integers have to be converted into double-base chain format. One example of algorithm converting an integer n into (2,3) double-base chain representation is shown in Algorithm 1.

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

Input: $n > 0$ and a_{max}, b_{max} [1] > 0
Output: 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 $b_1 \geq b_2 \geq \cdots \geq b_\ell$

```
d ← 1
while n > 0 do
    r ← 2^a 3^b, best approximation of n with 0 ≤ a ≤ a_max and 0 ≤ b ≤ b_max
    a_max ← a, b_max ← b
    print (d, a, b)
    if r > n then
        d ← −1
    end if
    n ← |n − r|
end while
```

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

2.3 Field Arithmetic Operations

Cohen, Miyaji, and Ono [3] 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 [6] showed that when point tripling is computed consecutively or when point doubling is computed after point tripling, some intermediate values from previous computations can be reused to help reducing one computation of field squaring.

For curves defined over prime field, cost of field squaring is less than that of field multiplication, or approximately $[s] = 0.8[m]$ where $[s]$ and $[m]$ are the cost of field squaring and field multiplication respectively. Longa and Miri [8] described how multiplication can be converted to squaring using the equality: $ab = \frac{1}{2}[(a+b)^2 - a^2 - b^2]$, where a, b are elements in prime field. They observed that if values a^2 and b^2 have already been calculated, ab can be computed from $(a + b)^2$ and use the equality above to obtain the value ab. We shall now refer to this method as *S-M trade-off*. This method allows three multiplications in point tripling formula and two multiplications in point doubling formula for curves defined over prime field to be converted to squaring.

3 Our Method

3.1 Elliptic Curves Defined over Prime Field

For curves defined over prime field in Jacobian coordinates, [6] explained that by computing point tripling *before* point doubling and reusing intermediate values, one computation of field squaring could be reduced. On the other hand, our study revealed that by computing point tripling *after* point doubling, two computations of field squaring could be reduced. Figure 1 illustrates our idea of switching point doubling and point tripling sequences.

We investigated point doubling and point tripling formulas for curves defined over prime field in Jacobian coordinates and analyzed values that have to be computed before reaching the final solutions of point doubling and point tripling. We shall now refer to those values as *intermediate values*.

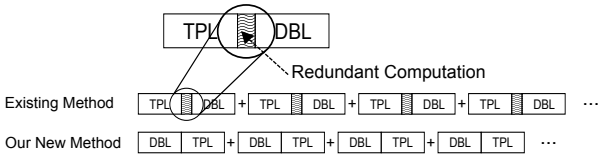

Figure 1: New sequences of field arithmetic operation to reduce redundant computations

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 1 summarizes intermediate values occurred during each step of computing point doubling. We categorized them into values obtained by field multiplication and field squaring.

Table 1: Intermediate values during point doubling on curves over $\mathbb{F}_p$ 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 1, aZ_1^4 needs to be computed in order to compute α_2. 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 2 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 2: Intermediate values during point tripling on curves over $\mathbb{F}_p$ 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
$X_3 = 8Y_2^2(\beta_3 - \alpha_3) + X_2\omega_3^2$	$Y_2^2(\beta_3 - \alpha_3), X_2\omega_3^2$	ω_3^2
$Y_3 = Y_2[4(\alpha_3 - \beta_3)(2\beta_3 - \alpha_3) - \omega_3^3]$	$(\alpha_3 - \beta_3)(2\beta_3 - \alpha_3), \omega_3^3,$ $Y_2[4(\alpha_3 - \beta_3)(2\beta_3 - \alpha_3) - \omega_3^3]$	–
$Z_3 = 2Z_2\omega_3$	$Z_2\omega_3$	–

In this work, we observed that the value $Z_2 = 2Y_1Z_1$ obtained when computing point doubling could be substituted into point tripling formula when computing θ_3, i.e., computing θ_3 from $3X_2^2 + 16aZ_1^4Y_1^4$. Consequently, the intermediate values aZ_1^4 and Y_1^4 occurred during point doubling can be reused to save two computations of field squaring. That is, θ_3 can be computed using only one squaring for X_2^2 and one multiplication for $(aZ_1^4)(Y_1^4)$. Comparing to direct computation from point tripling formula, it requires three squarings for X_2^2, Z_2^2, Z_2^4 and one multiplication for aZ_2^4. Table 3 shows our method of reusing intermediate values to reduce two computations of field squaring when point tripling is computed after point doubling.

Note that in [6], they considered the cost of point tripling to be $10[m] + 6[s]$. However, there exists *revised* tripling having cost $9[m] + 7[s]$. We derived our method from this revised tripling formula; hence, we compared our results to the previous results [6] of computing point

Table 3: Reducing two squarings by computing point tripling after point doubling for curves over $\mathbb{F}_p$ in Jacobian coordinates

Method	Operation	Intermediate Values	
		Multiplication	Squaring
Direct computation (Previous results)	θ_3=$3X_2^2 + aZ_2^4$	aZ_2^4	$\boldsymbol{X_2^2}, \boldsymbol{Z_2^2}, Z_2^4$
Reusing Values (Our results)	θ_3=$3X_2^2 + a[2Y_1Z_1]^4$ θ_3=$3X_2^2 + 2^4 \cdot aZ_1^4Y_1^4$	$(aZ_1^4)(Y_1^4)$	X_2^2

tripling *before* point doubling (w-TPL$^{\mathcal{J}}$/w'-DBL$^{\mathcal{J}}$) but considered as using revised tripling.

In Section 2.3, we mentioned S-M trade-off introduced by Longa and Miri [8]. This S-M trade-off changes multiplication into squaring plus three extra additions. By applying this idea of S-M trade-off, our results could further be improved.

Table 4 shows the cost comparison between previous results [6], namely, computing point tripling *before* point doubling (w-TPL$^{\mathcal{J}}$/w'-DBL$^{\mathcal{J}}$), and our results, namely, computing point tripling *after* point doubling (w'-DBL$^{\mathcal{J}}$/w-TPL$^{\mathcal{J}}$), both with and without S-M trade-off [8].

Table 4: Cost comparison between different orders of point doubling and point tripling

	Operation	Cost
Previous results [6]	(w-TPL$^{\mathcal{J}}$/w'-DBL$^{\mathcal{J}}$)	$(9w+4w')[m]+(6w+4w'+\mathbf{2})[s]$
Our results I	(w'-DBL$^{\mathcal{J}}$/w-TPL$^{\mathcal{J}}$ [1])	$(9w+4w')[m]+(6w+4w'+\mathbf{1})[s]$
Our results II	(w'-DBL$^{\mathcal{J}}$/w-TPL$^{\mathcal{J}}$ [2])	$(7w+3w'$-$1)[m]+(8w+5w'+2)[s]$

[1] without S-M trade-off [2] with S-M trade-off

3.2 Elliptic Curves Defined over Binary Field

We also investigated point doubling and point tripling formulas for curves defined over binary field in Jacobian coordinates and analyzed values that have to be computed before reaching the final solutions of point doubling and point tripling. Our study discovered that by reusing intermediate values, one computation of field squaring can be reduced for point tripling that either follows point tripling or point doubling. In other words, one field squaring can be omitted for each of the following point tripling in consecutive point triplings (w-TPL$^{\mathcal{J}}$), and the first point doubling that follows point tripling in consecutive point triplings then consecutive point doublings (w-TPL$^{\mathcal{J}}$/w'-DBL$^{\mathcal{J}}$).

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 5 summarizes intermediate values occurred during each step of computing point tripling on curves defined over binary field in Jacobian coordinates.

We can see from Table 5 that γ_3^2 is calculated when computing α_3. By using that fact that $[3]P_1 = (X_3, Y_3, Z_3) = (\alpha_3 Z_1^6, \beta_3 Z_1^9, \gamma_3 Z_1^3) = (\alpha_3, \beta_3, \gamma_3)$, calculating γ_3^2 is equivalent to calculating Z_3^2. If point tripling or point doubling is computed afterwards, the value Z_3^2 will appear again. In such cases, we can reused the value γ_3^2 so that computing Z_3^2 can be omitted.

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 6 summarizes intermediate values occurred 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 5: Intermediate values during point tripling on curves over $\mathbb{F}_{2^m}$ in Jacobian coordinates

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

$* \ (X_3, Y_3, Z_3) = (\alpha_3Z_1^6, \beta_3Z_1^9, \gamma_3Z_1^3) = (\alpha_3, \beta_3, \gamma_3)$

Table 6: Intermediate values during point doubling on curves over $\mathbb{F}_{2^m}$ in Jacobian coordinates

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

According to Table 6, the value Z_3^8 must be computed in order to obtain the value X_4. This indicated that the value Z_3^2 needs to be calculated. We found that we can reuse the value γ_3^2 so that we do not need to directly compute the value Z_3^2 at this step. Table 7 shows our method of reusing intermediate values to reduce one computation of field squaring.

Table 7: Reducing one squaring on curves over $\mathbb{F}_{2^m}$ in Jacobian coordinates by computing point doubling after point tripling

Method	Operation	Intermediate Values	
		Multiplication	Squaring
Direct computation (Previous results)	$X_4=X_3^4+a_6Z_3^8$	$a_6Z_3^8$	$X_3^2, X_3^4, \boldsymbol{Z_3^2}, Z_3^4, Z_3^8$
Reusing Values (Our results)	$\alpha_3=a_2\gamma_3^2+\omega_3(\omega_3+\gamma_3)+\theta_3^3$	$a_2\gamma_3^2, \omega_3(\omega_3+\gamma_3), \theta_3^3$	$\boldsymbol{\gamma_3^2}, \theta_3^2$
	$X_4=X_3^4+a_6Z_3^8$	$a_6Z_3^8$	$X_3^2, X_3^4, Z_3^4, Z_3^8 \quad (\gamma_3^2=Z_3^2)$

Table 8 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.

4 Experimental Results

We conducted several experiments using $10,000$ randomly chosen 256-bit integers. Those integers were converted to double-base *chain* of mixed power of two and three by the algorithm described in Algorithm 1 of Section 2.2 with a_{max} and b_{max} set to 150 and 100 respectively. These a_{max} and b_{max} values are related to the expansion length which effects scalar multiplication computing time. According to [6], 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.

Table 8: Cost for consecutive operations on curves defined over $\mathbb{F}_{2^m}$ in Jacobian coordinates

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 [6]

As describe in Section 2.3, general assumption for the cost of field squaring $[s]$ and field multiplication $[m]$ for curves defined over prime field is $[s] = 0.8[m]$. However, if we apply side-channel atomic block introduced in [2] to prevent simple side-channel analysis, we must consider $[s] = [m]$ instead. Our experiments included both assumptions of $[s] = [m]$ and $[s] = 0.8[m]$.

Table 9 shows cost comparison of curves defined over prime field in Jacobian coordinates between previous results, namely, consecutive point triplings followed by consecutive point doublings , and our results, namely, consecutive point doublings followed by consecutive point triplings.

Table 9: Cost comparison between previous and our results on curves over $\mathbb{F}_p$ in Jacobian coordinates

		Cost	$[s] = [m]$	$[s] = 0.8[m]$
ADD$^{\mathcal{J}}$	Previous results [1]	$1934.21[m] + 1337.19[s]$	3271.40	3003.96
	Our results I [2]	$1934.21[m] + 1278.84[s]$	3213.05	2957.28
	Our results II [3]	$1587.93[m] + 1625.12[s]$	3213.05	2888.02
ADD$^{\mathcal{J}+\mathcal{A}}$	Previous results [1]	$1700.81[m] + 1278.84[s]$	2979.65	2723.88
	Our results I [2]	$1700.81[m] + 1220.49[s]$	2921.30	2677.20
	Our results II [3]	$1354.53[m] + 1566.77[s]$	2921.30	2607.94

[1] w-TPL$^{\mathcal{J}}$/w'-DBL$^{\mathcal{J}}$ [6] [2] w'-DBL$^{\mathcal{J}}$/w-TPL$^{\mathcal{J}}$ [3] w'-DBL$^{\mathcal{J}}$/w-TPL$^{\mathcal{J}}$ with S-M trade-off

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

Together with the technique of converting field multiplication into field squaring proposed in [8], we obtained the reduction of 1.78% and 1.59% assuming $[s] = [m]$ and $[s] = 0.8[m]$ respectively if general addition was used. In case of using mixed addition, the reduction improved to 1.95% and 1.75%. We would like to emphasize that these reductions were the *second step* reductions, i.e., this results were compared to the ones that applied S-M trade off. If we compared to the results that did not apply any techniques, the total speed-up would be 4.25%.

For curves defined over binary field, it has been assumed that the cost of field squaring is very small compared to the cost of field multiplication and is neglected in many cases. However, with the current technology, this assumption becomes invalid in some situations. Consequently, the cost of field squaring tends to be taken into consideration. In our experiment, we assumed the cost of field squaring and field multiplication to be approximately $[s] = 0.1[m]$ according to [7].

Table 10 shows the cost comparison for computing consecutive point triplings followed by consecutive point doublings on curves defined over binary field in Jacobian coordinates between before and after applying our technique of reusing intermediate values.

According to the experimental results, with general addition, we achieved a reduction of 0.28%. If we use mixed addition, the reduction increased to 0.31%.

Table 10: Cost between previous and our results for computing w-TPL$^{\mathcal{J}}$/w'-DBL$^{\mathcal{J}}$ on curves over $\mathbb{F}_{2^m}$ in Jacobian coordinates

		Cost	$[s] = 0.1[m]$
ADD$^{\mathcal{J}}$	Previous results [6] [1]	2785.18$[m]$+1365.84$[s]$	2921.76
	Our results [2]	2785.18$[m]$+1282.86$[s]$	2913.46
ADD$^{\mathcal{J}+\mathcal{A}}$	Previous results [6] [1]	2494.60$[m]$+1365.84$[s]$	2631.19
	Our results [2]	2494.60$[m]$+1282.86$[s]$	2622.89

[1] without reusing intermediate values [2] with reusing intermediate values

5 Conclusion

We investigated sequences of field arithmetic operation when computing scalar multiplication and found redundancies among them. To remove those redundancies, our study indicated that rearrangement of those operations is required. That is, point doubling must be computed *before* point tripling for curves defined over prime field while point doubling must be computed *after* point tripling for curves defined over binary field in Jacobian coordinates. After rearranging those sequences, the concept of converting high-priced field multiplication into less expensive field squaring proposed by Longa and Miri [8] is still possible.

To precisely illustrate improvement achieved by applying our new method, several experiments using $10,000$ randomly chosen 256-bit integers have been conducted. By switching the sequences of point operation, we could reduce 1.95% of computational cost for curves defined over prime field in Jacobian coordinates assuming $[s] = 0.8[m]$ where $[s]$ and $[m]$ are the cost of field squaring and field multiplication respectively. Because field arithmetic operation is a building block for many computations including scalar multiplication, speeding up at field arithmetic level can further improve many applications in higher levels.

References

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

[2] B. Chevallier-Mames, M. Ciet, and M. Joye. Low-cost solutions for preventing simple side-channel analysis: Side-channel atomicity. *IEEE Trans. on Computers*, 53:760–768, 2004.

[3] 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*, 1998.

[4] V. Dimitrov, L. Imbert, and P. K. Mishra. Efficient and secure elliptic curve point multiplication using double-base chains. In *Proceedings of the 11th international conference on Theory and Application of Cryptology and Information Security*, 2005.

[5] V. Dimitrov, G. Jullien, and W. Miller. Theory and applications of the double-base number system. *IEEE Transactions on Computers*, 48:1098 –1106, 1999.

[6] V. S. Dimitrov, L. Imbert, and P. K. Mishra. The double-base number system and its application to elliptic curve cryptography. *Math. of Computation*, 77:1075–1104, 2008.

[7] 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*, 2000.

[8] P. Longa and A. Miri. Fast and flexible elliptic curve point arithmetic over prime fields. *IEEE Trans. on Computers*, 57:289–302, 2008.