# A Novel Pre-Computation Scheme of Window $\tau$NAF for Koblitz Curves

Wei Yu[1], Saud Al Musa[2], Guangwu Xu[2], and Bao Li[1]

[1] State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China
yuwei_1_yw@163.com, libao@iie.ac.cn
[2] Department of EE & CS, University of Wisconsin-Milwaukee, USA
{salmusa,gxu4uwm}@uwm.edu

**Abstract.** Let $E_a : y^2 + xy = x^3 + ax^2 + 1/\mathbb{F}_{2^m}$ be a Koblitz curve. The window $\tau$-adic nonadjacent-form (window $\tau$NAF) is currently the standard representation system to perform scalar multiplications on $E_a$ by utilizing the Frobenius map $\tau$. Pre-computation is an important part for the window $\tau$NAF. In this paper, we first introduce $\mu\bar{\tau}$-operations in lambda coordinates ($\mu = (-1)^{1-a}$ and $\bar{\tau}$ is the complex conjugate of the complex representation of $\tau$). Efficient formulas of $\mu\bar{\tau}$-operations are then derived and used in a novel pre-computation scheme to improve the efficiency of scalar multiplications using window $\tau$NAF. Our pre-computation scheme costs 7**M**+5**S**, 26**M**+16**S**, and 66**M**+36**S** for window $\tau$NAF with width 4, 5, and 6 respectively whereas the pre-computation with the state-of-the-art technique costs 11**M**+8**S**, 43**M**+18**S**, and 107**M**+36**S**. Experimental results show that our pre-computation is about 60% faster, compared to the best pre-computation in the literature. It also shows that we can save from 2.5% to 4.9% on the scalar multiplications using window $\tau$NAF with our pre-computation.

**Keywords:** Elliptic curve cryptography, Koblitz curve, Window $\tau$NAF, Pre-computation, Lambda coordinate.

## 1 Introduction

The family of Koblitz curves, proposed by Koblitz in [1], are non-supersingular curves defined over $\mathbb{F}_2$. This family of curves has a computational advantage that a faster scalar multiplication can be achieved by replacing point doubling with the Frobenius map. For each bit $a \in \{0, 1\}$, the Koblitz curves are given as

$$E_a : y^2 + xy = x^3 + ax^2 + 1.$$

These curves can be considered over the binary extension $\mathbb{F}_{2^m}$ as $m$ varies. Since $E_a(\mathbb{F}_2)$ is a subgroup of $E_a(\mathbb{F}_{2^m})$, one sees that $|E_a(\mathbb{F}_{2^m})| = |E_a(\mathbb{F}_2)| \cdot p$ for some positive integer $p$. It is of cryptographic interest to choose suitable $m$ that makes $p$ a prime. In the rest of our discussion, we just consider cases that $p$ is a prime. In the range of $160 < m < 600$, it is known that $\frac{|E_0(\mathbb{F}_{2^m})|}{|E_0(\mathbb{F}_2)|}$ is a

prime when $m = 233, 239, 277, 283, 349, 409, 571$, and $\frac{|E_1(\mathbb{F}_{2^m})|}{|E_1(\mathbb{F}_2)|}$ is a prime when $m = 163, 283, 311, 331, 347, 359$. Five Koblitz curves have been recommended by NIST [2] : K-163(a=1), K-233(a=0), K-283(a=0), K-409(a=0), and K-571(a=0).

The Frobenius map $\tau$ is an endomorphism of $E_a(\mathbb{F}_{2^m})$ defined by $\tau(x, y) = (x^2, y^2)$ and $\tau(\mathcal{O}) = \mathcal{O}$ where $\mathcal{O}$ is the point at infinity. Let $\mu = (-1)^{1-a}$, then for each point $P$ in $E_a(\mathbb{F}_{2^m})$,

$$\tau^2(P) + 2P = \mu\tau(P).$$

This means that $\tau$ can be interpreted as a complex number $\tau = \frac{\mu+\sqrt{-7}}{2}$ satisfying $\tau^2 - \mu\tau + 2 = 0$. The Euclidean domain $\mathbb{Z}[\tau] = \mathbb{Z} + \tau\mathbb{Z}$ can be identified as a set of endomorphisms of $E_a$ in the sense that $(g + h\tau)P = gP + h\tau(P)$.

Let $M$ be the main subgroup of $E_a(F_{2^m})$, namely the subgroup of order $p$. $M$ is an annihilating subgroup of $\delta = \frac{\tau^m-1}{\tau-1}$ in the sense that $\delta(P) = \mathcal{O}$ for every $P \in M$. We also note that $N(\delta) = p$ where $N$ is the norm function on $\mathbb{Z}[\tau]$ defined as $N(g + h\tau) = |g + h\tau|^2 = g^2 + \mu gh + 2h^2$. It is easy to see that for an integer $n$ and an element $\rho \in \mathbb{Z}[\tau]$, if $\rho \equiv n \pmod{\delta}$, then $\rho P = nP$ holds.

Koblitz [1] proposed a method of computing scalar multiplication $nP$ with $P$ from the main subgroup of a Koblitz curve by representing $n = \sum_{i=0}^{l-1} \epsilon_i \tau^i$ with $\epsilon_i \in \{0, 1\}$ and evaluating $\sum_{i=0}^{l-1} \epsilon_i \tau^i(P)$. In [3], Solinas further developed an extremely efficient window $\tau$NAF to compute $nP$. Refinements and extensions of Solinas' method were obtained by Blake, Murty and Xu [4, 5].

The procedure of window $\tau$NAF can be described as four steps [6].

1. Reduction. Find a suitable $\rho \in \mathbb{Z}[\tau]$ satisfying $\rho \equiv n \pmod{\delta}$.
2. Window $\tau$NAF with width $w$. We shall just consider the nontrivial case of $w \geq 3$. Let $I_w = \{1, 3, \ldots, 2^{w-1} - 1\}$. For each $i \in I_w$, we choose an element $c_i$ from the set $R_i = \{g + h\tau | g + h\tau \equiv i \pmod{\tau^w}, N(g + h\tau) < 2^w\}$, and construct the coefficient set $C = \{c_1, c_3, \ldots, c_{2^{w-1}-1}\}$. The window $\tau$NAF of $n$ is the following sparse $\tau$ expansion of its reduction $\rho$:

$$\rho = \sum_{i=0}^{l-1} \epsilon_i u_i \tau^i,$$

   where $\epsilon_i \in \{-1, 1\}$ and $u_i \in C \cup \{0\}$ with the property that any set $\{u_k, u_{k+1}, \ldots, u_{k+w-1}\}$ contains at most one nonzero element.
3. Pre-computation. Compute $Q_i = c_i P$ for each $i \in I_w$.
4. Computing $nP$. Employ Horner's algorithm to calculate $nP$ using window $\tau$NAF and pre-computation.

Pre-computation plays a significant role in improving the efficiency of scalar multiplications using window $\tau$NAF. Several ways of designing pre-computations have been proposed by Solinas [3]; Blake, Murty and Xu [5]; and Hankerson, Menezes, and Vanstone [7]. Recently, Trost and Xu [6] introduced an optimal pre-computation of window $\tau$NAF that improves previous results. However, the main objective of the pre-computation in [6] are its mathematically natural and

clean forms. The optimality is only based on the fact that it requires $2^{w-2}-1$ point additions and two evaluations of the Frobenius map $\tau$. It achieves an improvement on performance and provides a convenient structure for further work. Even though the use of $\lambda$-projective coordinate system was touched upon in [6], it did not get into the arithmetic detail to save field operations (such as field multiplication and squaring).

The $\lambda$-coordinates (lambda representation) system is a very efficient coordinate system for binary elliptic curves proposed by Oliveira, López, Aranha, and Rodríguez-Henríquez in [8]. These authors demonstrated that more savings can be achieved using $\lambda$-projective coordinates for point addition and doubling compared to other coordinates. For example, a full point addition requires 11 multiplications and 2 squarings (see [8]). For Koblitz curves, application of $\lambda$-projective coordinates in pre-computation for window $\tau$NAF has been briefly discussed in [6], together with efficient formulas for $P-\mu\tau(P)$, $P+\mu\tau(P)$ and $P-\mu\tau^2(P)$.

Because of its promising computational advantage, it is of great interest to consider the use of $\lambda$-projective coordinates in the window $\tau$NAF for Koblitz curves, especially for the pre-computation part with a new design. Now let us summarize the cost of existing pre-computation schemes in $\lambda$-projective coordinates for window width $w=4$, 5, and 6 (for $w=3$, $P-\mu\tau P$ is the only pre-computation). We write **I**, **M**, and **S** for the costs of an inversion, a multiplication, and a squaring in $\mathbb{F}_{2^m}$ respectively. The pre-computation scheme in [3] covers $w=4$, 5 only and their corresponding costs are 15**M**+12**S** and 44**M**+31**S**. In [7], $w=4$, 5, and 6 are considered and the corresponding costs are 15**M**+12**S**, 50**M**+29**S**, and 117**M**+63**S**. By efficient formulas for $P-\mu\tau(P)$, $P+\mu\tau(P)$ and $P-\mu\tau^2(P)$, the pre-computation scheme constructed in [6] has improved costs of 11**M**+8**S**, 43**M**+18**S**, and 107**M**+36**S** for $w=4$, 5, and 6.

**Our contributions** The main purpose of this work is to develop a more efficient way of calculating pre-computation for the window $\tau$NAF on Koblitz curves. By using $\lambda$-projective coordinates, our results show a great improvement over previous results. The main contributions are described as follows.

1. Let $\bar{\tau}=\mu-\tau$ be the complex conjugate of $\tau$ and $P=(X_P,\Lambda_P,Z_P)$. Both Avanzi, Dimitrov, Doche, and Sica [9] and Doche, Kohel, and Sica [10] use $\bar{\tau}$ in double-base representation to speeding up scalar multiplication. Inspired by their elegant results, we introduce a new radix $\mu\bar{\tau}$. Under this radix and $\lambda$-projective coordinates, our new formulas of $\mu\bar{\tau}P$ ($\mu\bar{\tau}$-operation) require 5**M**+3**S**, formulas of $(\mu\bar{\tau})^2P$ (re-$\mu\bar{\tau}$ operation) and $(\mu\bar{\tau})^iP, i\geq 3$ (re-re-$\mu\bar{\tau}$ operation) both require 3**M**+2**S**, formulas of $\mu\bar{\tau}P$ and $(\mu\bar{\tau})^2P$ ($\mu\bar{\tau}\&(\mu\bar{\tau})^2$-affine operation) require 4**M**+3**S**. It is noted that one point addition is necessary for computing each pre-computation point $Q_i$, $i\in\{3,5,\ldots,2^{w-1}-1\}$ as it has been proved in [6]. We use $\mu\bar{\tau}$-operations to replace point additions or mixed additions in pre-computation scheme. As one full addition requires 11**M**+2**S** and one mixed addition requires 8**M**+2**S**, these point operations proposed in this work all save quite a few field operations.

2. Trost and Xu's pre-computation is unified (without treating $a = 0$ and $a = 1$ separately). For example, consider $w = 4$, we have $Q_5 = -P + \mu\tau P$, $Q_7 = P + \mu\tau P$, $Q_3 = -P + \tau^2 P$. Also cases $w = 5$, 6 can be unified. We propose a plane search to generate $R_i$ whose elements are with the form of $g + h\mu\tau$. Since $Q_i = c_i P, c_i \in R_i$ for each $i \in I_w$, $\{Q_i, i \in I_w\}$ is a unified pre-computation.
3. To take full advantage of our $\mu\bar{\tau}$-operations, we choose one suitable $c_i \in R_i$ for each $i \in I_w$ which is generated by the plane search. A novel unified pre-computation scheme is developed to save more field operations, e.g., it requires 7**M**+5**S**, 26**M**+16**S**, and 66**M**+36**S** for the cases of $w = 4$, 5, and 6 respectively.
4. The costs of Solinas' pre-computation scheme; Hankerson, Menezes, and Vanstone's pre-computation scheme; Trost and Xu's pre-computation scheme; and our pre-computation scheme are shown in Table 1. Theoretical analysis shows that our pre-computation scheme is about 110% faster than Solinas' pre-computation scheme for $w = 4$ and 70% for $w = 5$. It is about 110% faster than Hankerson, Menezes, and Vanstone's pre-computation scheme for $w = 4$, 90% for $w = 5$, and 77% for $w = 6$. It is about 60% faster than Trost and Xu's pre-computation scheme for $w = 4$, 5, and 6. The practical implementations are consistent with its theoretical analysis.
5. To get a higher efficiency of scalar multiplication, we perform it using the pre-computation in $\lambda$-coordinates in which mixed addition is fully utilized. We employ Montgomery trick [7] to translate the points in $\lambda$-projective coordinates to that in $\lambda$-coordinates. Both theoretical analysis and experimental results show that scalar multiplications using our scheme are faster than those using previous schemes. On K-163, K-233, K-283, K-409, and K-571, scalar multiplications using our pre-computation scheme are 2.8%, 4.2%, 3.1%, 2.5%, and 4.9% faster than those using Trost and Xu's pre-computation scheme respectively.

**Table 1.** Cost of pre-computations using $\lambda$-coordinates

| Window size | 4 | 5 | 6 |
|---|---|---|---|
| Solinas | 15**M**+12**S** | 44**M**+31**S** | - |
| Hankerson, Menezes, Vanstone | 15**M**+12**S** | 50**M**+29**S** | 117**M**+63**S** |
| Trost, Xu | 11**M**+8**S** | 43**M**+18**S** | 107**M**+36**S** |
| Ours | 7**M**+5**S** | 26**M**+16**S** | 66**M**+36**S** |

This paper is organized as follows. In Section 2, we present previous pre-computation schemes of window $\tau$NAF for Koblitz curves. In Section 3, we propose new formulas of $P \pm Q$ and $\mu\bar{\tau}$-operations. In Section 4, we design a novel pre-computation. In Section 5, scalar multiplications using different pre-computation schemes are analyzed. In Section 6, we compare our pre-computation scheme to other pre-computation schemes in experimental implementations. Finally, we conclude this paper.

## 2 Preliminary

We shall include some technical preparation and existing designs of pre-computations in this section.

### 2.1 Montgomery Trick

Montgomery trick [7] computes simultaneously the inversions of $n$ elements. It requires one inversion and $3(n-1)$ multiplications, and is usually useful when $\mathbf{I}/\mathbf{M}>3$. This trick is powerful to translate points in projective coordinates to

**Algorithm 1** Montgomery trick

**Input:** $a_1, a_2, \ldots, a_n$
**Output:** $b_1 = {a_1}^{-1}, b_2 = {a_2}^{-1}, \ldots, b_n = {a_n}^{-1}$
**Computation**

1. $c_1 \leftarrow a_1$
2. for $i \leftarrow 2$ to $n$
   $c_i \leftarrow c_{i-1} \cdot a_i$
3. $d \leftarrow c_n^{-1}$
4. for $i \leftarrow n$ to 2
   $b_i \leftarrow c_{i-1} \cdot d$
   $d \leftarrow a_i \cdot d$
5. $b_1 \leftarrow d$
6. output $b_i$

points in affine coordinates. In this work, we use Montgomery trick to translate points in $\lambda$-projective coordinates to points in $\lambda$-coordinates. For $n$ points $(X_i, \Lambda_i, Z_i), 1 \leq i \leq l$, we use Montgomery trick to compute $Z_i^{-1}$, and then compute $(x_i = \frac{X_i}{Z_i}, \lambda_i = \frac{\Lambda_i}{Z_i})$. This Montgomery trick translating $n$ points in $\lambda$-projective coordinates to these in $\lambda$-coordinates requires $\mathbf{I}+(5n-3)\mathbf{M}$. When points in $\lambda$-projective coordinates are converted to points in $\lambda$-coordinates, we replace full point addition with mixed point addition to get a higher efficiency of scalar multiplication.

### 2.2 Hensel's Lifting Algorithm

In the later discussion, we need a convenient criterion to determine whether $\tau^w | g + h\tau$ holds in $\mathbb{Z}[\tau]$. This can be done by using Lucas sequence as in [3]. But we shall take the approach suggested in [5] based on Hensel's lifting procedure [11].

Let $f(x) = x^2 - \mu x + 2$ and $s_n = b_1 2 + b_2 2^2 + \ldots + b_{n-1} 2^{n-1}, 0 \leq b_j \leq 1$ be the $n$-th 2-adic approximation of a zero of $f(x)$. From $f(s_i) \equiv 0 \pmod{2^i}$, we have $s_1 = 0$, $s_2 = s_1 + b_1 \cdot 2$, $\ldots$, $s_{n+1} = s_n + b_n 2^n$. Thus we only need to calculate

$s_{n-1}$ and $b_n$ to get $s_n$. Since $f(s_{n+1}) = f(s_n) + 2b_n 2^n s_n + b_n^2 2^{2n} - ub_n 2^n$, i.e., $f(s_n) + b_n 2^{n+1} s_n + b_n^2 2^{2n} - \mu b_n 2^n \equiv 0 \pmod{2^{n+1}}$, $b_n \equiv \frac{\mu f(s_n)}{2^n} \pmod 2$ holds true. The precise Hensel's lifting algorithm is shown as Algorithm 2 [6].

**Algorithm 2** The $n$th 2-adic approximation

**Computation**

1. $s_1 \leftarrow <>$
2. for $i \leftarrow 1$ to $n-1$
   $b_i \leftarrow \frac{f(s_i)}{2^i}\mu \pmod 2$ //since $2^i | f(s_i)$
   $s_{i+1} \leftarrow s_i + b_i 2^i$
3. output $s_i$

From Algorithm 2, we calculate $s_2 = 2\mu$, $s_3 = 6\mu$, $s_4 = 6\mu$, $s_5 = 6\mu$, $s_6 = 38\mu$, $s_7 = 38\mu$, $s_8 = 166\mu$, $s_9 = 422\mu$, $s_{10} = 934\mu, \ldots$. When $n \geq 2$, $s_n \equiv 0 \pmod 2$.

It has been proved in [5] that for each positive integer $w$,

$$\tau^w | g + h\tau \Leftrightarrow 2^w | g + hs_w. \tag{1}$$

### 2.3 Previous Pre-Computation Schemes

We will consider efficacy of pre-computation schemes in the context of $\lambda$-projective coordinates. This type of coordinate system was proposed by Oliveira, López, Aranha, and Rodríguez-Henríquez in [8] for elliptic curves over binary fields. Given an affine point $P = (x, y)$ on an elliptic curve $E/\mathbb{F}_{2^m}$, its lambda representation is $(x, \lambda)$ with $\lambda = x + \frac{y}{x}$. This can be converted to the projective coordinates in the usual manner.

It has been demonstrated that the computation of point addition, mixed point addition[3], and $\tau(P)$ using $\lambda$-projective coordinates [8] are usually faster than these using LD coordinates [12]. In the rest of our discussion, point operations using $\lambda$-projective coordinates are considered. We neglect the cost of a field addition since it involves only bitwise XORs. We will use the following cost calculations: one mixed point addition costs $8\mathbf{M}+2\mathbf{S}$ (the cost of mixed point addition is $5\mathbf{M}+2\mathbf{S}$ when $Z$-components of both two summands can be set to 1, denoted by mixed addition with $Z = 1$), one full point addition costs $11\mathbf{M}+2\mathbf{S}$. Furthermore, evaluation of $\tau(P)$ costs $3\mathbf{S}$ in general ($\tau$-operation) and $2\mathbf{S}$ when the $Z$-coordinate of $P$ is 1 ($\tau$-affine operation). When the $Z$-coordinate of $P$ is 1, Trost and Xu introduced formulas for $P - \mu\tau P$, $P + \mu\tau P$ (utilizing some come values for computing $P - \mu\tau P$) which require $2\mathbf{M}+2\mathbf{S}$, $4\mathbf{M}+3\mathbf{S}$ respectively, and also provided efficient formulas for $P - \mu\tau^2 P$ only requiring $5\mathbf{M}+3\mathbf{S}$ if $P + \mu\tau P$ had been calculated.

[3] In this case, the $Z$-component of one of the summand can be set to 1.

**Solinas' pre-computation** Solinas [3] suggested an efficient design of the pre-computation and gave an example as shown in Table 2. It requires 3 mixed point additions (include 3 mixed additions with $Z = 1$) and 3 $\tau$-affine operations for the case of $w = 4$; 7 mixed point additions (include 4 mixed additions with $Z = 1$), 4 $\tau$-affine operations, and 3 $\tau$-operations for the case of $w = 5$. The costs are 15**M**+12**S** and 44**M**+31**S** respectively.

**Table 2.** Pre-computation scheme for $w = 4, 5$ in [3] ($a$=1)

| | | |
|---|---|---|
| w=4 | | |
| $Q_3 = -P + \tau^2 P$ | $Q_5 = P + \tau^2 P$ | $Q_7 = -P - \tau^3 P$ |
| w=5 | | |
| $Q_3 = -P + \tau^2 P$ | $Q_5 = P + \tau^2 P$ | $Q_7 = -P - \tau^3 P$ |
| $Q_9 = P - \tau^3 Q_5$ | $Q_{11} = -\tau^2 Q_5 - P$ | $Q_{13} = -\tau^2 Q_5 + P$ |
| $Q_{15} = -P + \tau^4 P$ | | |

**Hankerson, Menezes, and Vanstone's pre-computation** Hankerson, Menezes, and Vanstone [7] presented an improved design of pre-computation as shown in Table 3. For $w = 4$, this pre-computation scheme requires 3 mixed point additions (include 3 mixed additions with $Z = 1$) and 3 $\tau$-affine operations. It takes 1 full point addition, 6 mixed point additions (include 3 mixed additions with $Z = 1$), 3 $\tau$-affine operations, and 3 $\tau$-operations for the case of $w = 5$; 3 full point additions, 12 mixed point additions (include 4 mixed additions with $Z = 1$), 3 $\tau$-affine operations, and 9 $\tau$-operations for the case of $w = 6$. So the costs for the three cases are 15**M**+12**S**, 50**M**+29**S**, and 117**M**+63**S** respectively.

**Table 3.** Pre-computation scheme for $w = 4, 5, 6$ in [7] ($a$=1)

| | | |
|---|---|---|
| w=4 | | |
| $Q_3 = -P + \tau^2 P$ | $Q_5 = P + \tau^2 P$ | $Q_7 = -P - \tau^3 P$ |
| w=5 | | |
| $Q_3 = -P + \tau^2 P$ | $Q_5 = P + \tau^2 P$ | $Q_7 = -P - \tau^3 P$ |
| $Q_9 = P - \tau^3 Q_5$ | $Q_{11} = -\tau^2 Q_5 - P$ | $Q_{13} = -\tau^2 Q_5 + P$ |
| $Q_{15} = -Q_5 + \tau^2 Q_5$ | | |
| w=6 | | |
| $Q_{23} = -P - \tau^3 P$ | $Q_{25} = P - \tau^3 P$ | $Q_{27} = -P - \tau^2 P$ |
| $Q_{29} = P - \tau^2 P$ | $Q_3 = \tau^2 Q_{25} - P$ | $Q_5 = \tau^2 Q_{25} + P$ |
| $Q_7 = -\tau^3 Q_{27} - P$ | $Q_9 = -\tau^3 Q_{27} + P$ | $Q_{11} = \tau^2 Q_{27} - P$ |
| $Q_{13} = \tau^2 Q_{27} + P$ | $Q_{15} = -\tau^2 Q_{27} + Q_{27}$ | $Q_{17} = -\tau^2 Q_{27} + Q_{29}$ |
| $Q_{19} = -\tau^2 Q_3 - P$ | $Q_{21} = \tau^2 Q_{29} + P$ | $Q_{31} = \tau^2 Q_{25} + Q_{27}$ |

**Trost and Xu's pre-computation** Trost and Xu [6] proposed a mathematically natural and clean form of pre-computation, it requires the least number of point additions and $\tau$ evaluations. We include here their pre-computation scheme and adjust the calculation order to achieve its best computing speed for $w = 4$, 5, and 6 in Table 4. Specifically, their pre-computation scheme requires 1 $(P - \tau P)$-operation, 1 $(P + \tau P)$-operation, and 1 $(P - \tau^2 P)$-operation for

$w = 4$; 1 $(P - \tau P)$-operation, 1 $(P + \tau P)$-operation, 1 $(P - \tau^2 P)$-operation, 4 mixed point additions and 1 $\tau$-affine operation for $w = 5$; 1 $(P - \tau P)$-operation, 1 $(P + \tau P)$-operation, 1 $(P - \tau^2 P)$-operation, 12 mixed point additions and 2 $\tau$-affine operations for $w = 6$. The costs are $11\mathbf{M}+8\mathbf{S}$, $43\mathbf{M}+18\mathbf{S}$, and $107\mathbf{M}+36\mathbf{S}$ respectively.

**Table 4.** Pre-computation scheme for $w = 4, 5, 6$ in [6] ($a$=1)

| | | |
|---|---|---|
| w=4 | | |
| $Q_5 = -P + \tau P$ | $Q_7 = P + \tau P$ | $Q_3 = -P + \tau^2 P$ |
| w=5 | | |
| $Q_5 = -P + \tau P$ | $Q_7 = P + \tau P$ | $Q_3 = -P + \tau^2 P$ |
| $Q_9 = Q_3 + \tau P$ | $Q_{11} = Q_5 + \tau P$ | $Q_{13} = Q_7 + \tau P$ |
| $Q_{15} = -Q_{11} - \tau P$ | | |
| w=6 | | |
| $Q_{27} = P - \tau P$ | $Q_{25} = -P - \tau P$ | $Q_{29} = P - \tau^2 P$ |
| $Q_3 = Q_{29} + \tau P$ | $Q_9 = -Q_{29} + \tau P$ | $Q_{31} = Q_3 - \tau^2 P$ |
| $Q_5 = Q_{31} + \tau P$ | $Q_7 = -Q_{31} + \tau P$ | $Q_{11} = -Q_{27} + \tau P$ |
| $Q_{13} = -Q_{25} + \tau P$ | $Q_{15} = -Q_{11} - \tau P$ | $Q_{17} = -Q_9 - \tau P$ |
| $Q_{19} = -Q_7 - \tau P$ | $Q_{21} = -Q_{17} + \tau P$ | $Q_{23} = -Q_3 - \tau P$ |

It is remarked that [6] only discussed the use of $\lambda$-projective coordinates in a brief manner. The authors did not get into field arithmetic details to speed up the pre-computation. Our main objective of this paper is to further explore the use of $\lambda$-projective coordinates through some novel arrangement of pre-computation and efficient formulas to achieve a great saving in terms of the number of field multiplications and squarings. In the next section, we will propose new efficient formulas using $\lambda$-coordinates to design an efficient pre-computation scheme.

## 3 New Formulas using $\lambda$-coordinates

Let $P = (x_P, \lambda_P)$, $\lambda_P = x_P + \frac{y_P}{x_P}$. The $\lambda$-coordinates of $-P$ are $(x_P, \lambda_P + 1)$. The $\lambda$-projective coordinates of $P$ are $(X_P, \Lambda_P, Z_P)$ where $x_P = \frac{X_P}{Z_P}$ and $\lambda_P = \frac{\Lambda_P}{Z_P}$. We have $\tau(x_P, \lambda_P) = (x_P^2, \lambda_p^2)$ and $\tau(X_P, \Lambda_P, Z_P) = (X_P^2, \Lambda_P^2, Z_P^2)$. Let $P = (x_P, \lambda_P)$ and $Q = (x_Q, \lambda_Q)$. Point addition $P + Q = (x_{P+Q}, \lambda_{P+Q})$ was given in [8] as

$$\begin{cases} x_{P+Q} = \frac{x_P x_Q}{(x_P + x_Q)^2}(\lambda_P + \lambda_Q), \\ \lambda_{P+Q} = \frac{x_Q (x_{P+Q} + x_P)^2}{x_{P+Q} x_P} + \lambda_P + 1. \end{cases} \tag{2}$$

### 3.1 New Formulas for $P \pm Q$

$P \pm Q$ are first proposed to improve the efficiency of pre-computation on elliptic curves over a prime field [13]. We will show its formulas over a binary field in $\lambda$-projective coordinates by Theorem 1.

**Theorem 1** *Let $P = (x_P, \lambda_P)$, $Q = (X_Q, \Lambda_Q, Z_Q)$, $P \neq \pm Q$. $P \pm Q$ ($(P \pm Q)$-operation) can be computed as Algorithm 3 at the cost of* $12\mathbf{M}+5\mathbf{S}$.

**Algorithm 3** $(P \pm Q)$-operation

**Input:** $P = (x_P, \lambda_P)$, $Q = (X_Q, \Lambda_Q, Z_Q)$
**Output:** $P + Q = (X_{P+Q}, \Lambda_{P+Q}, Z_{P+Q})$, $P - Q = (X_{P-Q}, \Lambda_{P-Q}, Z_{P-Q})$
**Computation**

| | |
|---|---|
| 1. $A = \lambda_P Z_Q + \Lambda_Q$ | **M** |
| 2. $B = (x_P Z_Q + X_Q)^2$ | **M+S** |
| 3. $C = X_Q Z_Q$ | **M** |
| 4. $D = x_P C$ | **M** |
| 5. $X_{P+Q} = A^2 D$ | **M+S** |
| 6. $Z_{P+Q} = BAZ_Q$ | **2M** |
| 7. $\Lambda_{P+Q} = (AX_Q + B)^2 + Z_{P+Q}(\lambda_P + 1)$ | **2M+S** |
| 8. $X_{P-Q} = X_{P+Q} + DZ_Q^2$ | **M+S** |
| 9. $Z_{P-Q} = Z_{P+Q} + BZ_Q^2$ | **M** |
| 10. $\Lambda_{P-Q} = \Lambda_{P+Q} + C^2 + BZ_Q^2(\lambda_P + 1)$ | **M+S** |

*Proof.* By Equation (2), mixed addition $P + Q = (X_{P+Q}, \Lambda_{P+Q}, Z_{P+Q})$ was given in [8] as

$$\begin{aligned} A =& \lambda_P Z_Q + \Lambda_Q, \\ B =& (x_P Z_Q + X_Q)^2, \\ X_{P+Q} =& x_P A X_Q A Z_Q, \\ Z_{P+Q} =& BAZ_Q, \\ \Lambda_{P+Q} =& (AX_Q + B)^2 + Z_{P+Q}(\lambda_P + 1). \end{aligned}$$

These formulas of mixed addition require 8**M**+2**S**. If $Z_Q = 1$, the cost of mixed addition is 5**M**+2**S**.

Notice that $-Q = (X_Q, \Lambda_Q + Z_Q, Z_Q)$. The formulas of $P \pm Q$ can be computed as follows.

$$\begin{aligned} A =& \lambda_P Z_Q + \Lambda_Q, \\ B =& (x_P Z_Q + X_Q)^2, \\ C =& X_Q Z_Q, \\ D =& x_P C, \\ X_{P+Q} =& A^2 D, \\ Z_{P+Q} =& BAZ_Q, \\ \Lambda_{P+Q} =& (AX_Q + B)^2 + Z_{P+Q}(\lambda_P + 1), \\ X_{P-Q} =& X_{P+Q} + DZ_Q^2, \\ Z_{P-Q} =& Z_{P+Q} + BZ_Q^2, \\ \Lambda_{P-Q} =& \Lambda_{P+Q} + C^2 + BZ_Q^2(\lambda_P + 1). \end{aligned}$$

These formulas of computing $P \pm Q$ are shown as Algorithm 3 requiring 12**M**+5**S**. ∎

Since individually computing $P+Q$ and $P-Q$ requires $16\mathbf{M}+4\mathbf{S}$, our formulas of $P \pm Q$ save $4\mathbf{M}-\mathbf{S}$. When $\tau P = (x_P^2, \lambda_P^2)$ is given, the cost of computing $P \pm \mu\tau Q$ are $12\mathbf{M}+5\mathbf{S}$.

Using new formulas of $P \pm Q$, Solinas' pre-computation scheme saves $4\mathbf{M}-\mathbf{S}$ for $w = 4$ and $8\mathbf{M}-2\mathbf{S}$ for $w = 5$; Hankerson, Menezes, and Vanstone's pre-computation scheme saves $4\mathbf{M}-\mathbf{S}$ for $w = 4$, $8\mathbf{M}-2\mathbf{S}$ for $w = 5$, and $20\mathbf{M}-5\mathbf{S}$ for $w = 6$; Trost and Xu's pre-computation scheme saves $8\mathbf{M}-2\mathbf{S}$ for $w = 6$. These formulas of computing $P \pm Q$ are also used in the design of our pre-computation scheme.

### 3.2 Formulas for $\mu\bar{\tau}$-operations

Avanzi, Dimitrov, Doche, and Sica [9] first introduce $\bar{\tau}$ to improve the efficiency of scalar multiplication. They noticed that $2 = \tau\bar{\tau}$ and compute $\bar{\tau}P$ requiring a point doubling and three square roots. Doche, Kohel, and Sica [10] propose a new efficient way to compute $\bar{\tau}P$ which induces a speedup on the scalar multiplication using double-base representation over 15% in LD coordinates. Motivated by their work, we introduce a new radix $\mu\bar{\tau}$ to speed up pre-computation stage of scalar multiplication using window $\tau$NAF. To present some new efficient formulas for $\mu\bar{\tau}$-operations, we propose formulas of $\mu\bar{\tau}P$ in $\lambda$-coordinates shown by Lemma 1.

**Lemma 1** *Let $P = (x_P, \lambda_P)$. $\mu\bar{\tau}P$ can be computed as*

$$\mu\bar{\tau}P = \left(\frac{x_P^2+1}{x_P}, \frac{x_P^2}{x_P^2+1} + \lambda_P\right).$$

*Proof.* Notice that $-\mu\tau P = (x_P^2, \lambda_P^2 + a)$ and $\mu\bar{\tau} = 1 - \mu\tau$. In $\lambda$-coordinates, $E_a$ can be represented as $\lambda^2 + \lambda + a = \frac{x^4+1}{x^2}$. By Equation (2) and $\mu\bar{\tau}P = P - \mu\tau P$, we have

$$\begin{aligned}
x_{\mu\bar{\tau}P} =& \frac{x_P x_{-\mu\tau P}}{(x_P + x_{-\mu\tau P})^2}(\lambda_P + \lambda_{-\mu\tau P}) \\
=& \frac{x_P}{1+x_P^2}(\lambda_P^2 + \lambda_P + a) \\
=& \frac{x_P^2+1}{x_P}, \\
\lambda_{\mu\bar{\tau}P} =& \frac{x_{-\mu\tau P}(x_{\mu\bar{\tau}P} + x_P)^2}{x_{\mu\bar{\tau}P} x_P} + \lambda_P + 1 \\
=& \frac{x_P^2}{1+x_P^2} + \lambda_P.
\end{aligned}$$

■

**Theorem 2** *Let $P = (X_P, \Lambda_P, Z_P)$. $\mu\bar{\tau}P$ ($\mu\bar{\tau}$-operation), $(\mu\bar{\tau})^2P$ (re-$\mu\bar{\tau}$ operation), and $(\mu\bar{\tau})^iP, i \geq 3$ (re-re-$\mu\bar{\tau}$ operation) can be computed as Algorithms 4, 5, and 6 at the cost of $5\mathbf{M}+3\mathbf{S}$, $3\mathbf{M}+2\mathbf{S}$, and $3\mathbf{M}+2\mathbf{S}$ respectively.*

**Algorithm 4** $\mu\bar{\tau}$-operation

**Input:** $P = (X_P, \Lambda_P, Z_P)$
**Output:** $\mu\bar{\tau}P = (X_{\mu\bar{\tau}P}, \Lambda_{\mu\bar{\tau}P}, Z_{\mu\bar{\tau}P})$, $\alpha = X_P Z_P$
**Computation**

| | |
|---|---|
| 1. $\alpha = X_P Z_P$ | **M** |
| 2. $A_1 = X_P^2 + Z_P^2$ | **2S** |
| 3. $X_{\mu\bar{\tau}P} = A_1^2$ | **S** |
| 4. $\Lambda_{\mu\bar{\tau}P} = \alpha X_P^2 + X_P \Lambda_P A_1$ | **3M** |
| 5. $Z_{\mu\bar{\tau}P} = A_1 \alpha$ | **M** |

**Algorithm 5** Re-$\mu\bar{\tau}$ operation

**Input:** $\alpha = X_P Z_P$, $\mu\bar{\tau}P = (X_{\mu\bar{\tau}P}, \Lambda_{\mu\bar{\tau}P}, Z_{\mu\bar{\tau}P})$ computed as Algorithm 4
**Output:** $(\mu\bar{\tau})^2 P = (X_{(\mu\bar{\tau})^2 P}, \Lambda_{(\mu\bar{\tau})^2 P}, Z_{(\mu\bar{\tau})^2 P})$
**Computation**

| | |
|---|---|
| 1) $A_2 = X_{\mu\bar{\tau}P} + \alpha^2$ | **S** |
| 2) $X_{(\mu\bar{\tau})^2 P} = A_2^2$ | **S** |
| 3) $\Lambda_{(\mu\bar{\tau})^2 P} = X_{\mu\bar{\tau}P} Z_{\mu\bar{\tau}P} + \Lambda_{\mu\bar{\tau}P} A_2$ | **2M** |
| 4) $Z_{(\mu\bar{\tau})^2 P} = Z_{\mu\bar{\tau}P} A_2$ | **M** |

**Algorithm 6** Re-re-$\mu\bar{\tau}$ operation

**Input:** A given $i$ ($i \geq 3$), $(\mu\bar{\tau})^{i-1} P = (X_{(\mu\bar{\tau})^{i-1} P}, \Lambda_{(\mu\bar{\tau})^{i-1} P}, Z_{(\mu\bar{\tau})^{i-1} P})$ (computed as Algorithm 5 if $i = 3$, as Algorithm 6 if $i > 3$), $Z_{(\mu\bar{\tau})^{i-2} P}$ (computed as Algorithm 4 if $i = 3$, as Algorithm 5 if $i = 4$, as Algorithm 6 if $i > 4$)
**Output:** $(\mu\bar{\tau})^i P = (X_{(\mu\bar{\tau})^i P}, \Lambda_{(\mu\bar{\tau})^i P}, Z_{(\mu\bar{\tau})^i P})$
**Computation**

| | |
|---|---|
| 1) $A_i = Z_{(\mu\bar{\tau})^{i-2} P}^2 + X_{(\mu\bar{\tau})^{i-1} P}$ | **S** |
| 2) $X_{(\mu\bar{\tau})^i P} = A_i^2$ | **S** |
| 3) $\Lambda_{(\mu\bar{\tau})^i P} = X_{(\mu\bar{\tau})^{i-1} P} Z_{(\mu\bar{\tau})^{i-1} P} + \Lambda_{(\mu\bar{\tau})^{i-1} P} A_i$ | **2M** |
| 4) $Z_{(\mu\bar{\tau})^i P} = Z_{(\mu\bar{\tau})^{i-1} P} A_i$ | **M** |

*Proof.* 1. By $\mu\bar{\tau}P = \left(\frac{x_P^2+1}{x_P}, \frac{x_P^2}{x_P^2+1} + \lambda_P\right)$ in Lemma 1, we have

$$\mu\bar{\tau}P = \left(\frac{(\frac{X_P}{Z_P})^2+1}{\frac{X_P}{Z_P}}, \frac{(\frac{X_P}{Z_P})^2}{(\frac{X_P}{Z_P})^2+1} + \frac{\Lambda_P}{Z_P}\right).$$

Then

$$\begin{aligned}
\alpha =& X_P Z_P,\\
A_1 =& X_P^2 + Z_P^2,\\
X_{\mu\bar{\tau}P} =& A_1^2,\\
\Lambda_{\mu\bar{\tau}P} =& \alpha X_P^2 + X_P \Lambda_P A_1,\\
Z_{\mu\bar{\tau}P} =& A_1 \alpha.
\end{aligned}$$

This is described as Algorithm 4.

2. Notice that $(\mu\bar{\tau})^2 P = \mu\bar{\tau}(\mu\bar{\tau}P)$,

$$(\mu\bar{\tau})^2 P = \left(\frac{x_{\mu\bar{\tau}P}^2+1}{x_{\mu\bar{\tau}P}}, \frac{x_{\mu\bar{\tau}P}^2}{x_{\mu\bar{\tau}P}^2+1} + \lambda_{\mu\bar{\tau}P}\right).$$

Motivated by that some values for computing $P - \mu\tau P$ are utilized to compute $P - \mu\tau P$ in [6], some values for computing $\mu\bar{\tau}P$ are used to compute $(\mu\bar{\tau})^2 P$. Let $\mu\bar{\tau}P = (X_{\mu\bar{\tau}P}, \Lambda_{\mu\bar{\tau}P}, Z_{\mu\bar{\tau}P})$ be computed as Algorithm 4 where $x_{\mu\bar{\tau}P} = \frac{A_1}{\alpha}$. We have

$$\begin{aligned}
(\mu\bar{\tau})^2 P &= \left(\frac{(\frac{A_1}{\alpha})^2+1}{\frac{A_1}{\alpha}}, \frac{(\frac{A_1}{\alpha})^2}{(\frac{A_1}{\alpha})^2+1} + \frac{\Lambda_{\mu\bar{\tau}P}}{Z_{\mu\bar{\tau}P}}\right)\\
&= \left(\frac{A_1^2+\alpha^2}{A_1\alpha}, \frac{A_1^2}{A_1^2+\alpha^2} + \frac{\Lambda_{\mu\bar{\tau}P}}{Z_{\mu\bar{\tau}P}}\right)\\
&= \left(\frac{X_{\mu\bar{\tau}P}+\alpha^2}{Z_{\mu\bar{\tau}P}}, \frac{X_{\mu\bar{\tau}P}}{X_{\mu\bar{\tau}P}+\alpha^2} + \frac{\Lambda_{\mu\bar{\tau}P}}{Z_{\mu\bar{\tau}P}}\right).
\end{aligned}$$

Then $(\mu\bar{\tau})^2(P)$ can be computed as

$$\begin{aligned}
A_2 =& X_{\mu\bar{\tau}P} + \alpha^2,\\
X_{(\mu\bar{\tau})^2 P} =& A_2^2,\\
\Lambda_{(\mu\bar{\tau})^2 P} =& X_{\mu\bar{\tau}P} Z_{\mu\bar{\tau}P} + \Lambda_{\mu\bar{\tau}P} A_2,\\
Z_{(\mu\bar{\tau})^2 P} =& Z_{\mu\bar{\tau}P} A_2.
\end{aligned}$$

It is shown as Algorithm 5.

3. When $i \geq 3$, $(\mu\bar{\tau})^i(P) = \mu\bar{\tau}((\mu\bar{\tau})^{i-1}(P))$.

$$(\mu\bar{\tau})^i P = \left(\frac{x_{(\mu\bar{\tau})^{i-1}P}^2+1}{x_{(\mu\bar{\tau})^{i-1}P}}, \frac{x_{(\mu\bar{\tau})^{i-1}P}^2}{x_{(\mu\bar{\tau})^{i-1}P}^2+1} + \lambda_{(\mu\bar{\tau})^{i-1}P}\right).$$

Some values of calculating $(\mu\bar{\tau})^{i-1}P$ are used to calculate $(\mu\bar{\tau})^{i}P$. When $i=3$, we compute $x_{(\mu\bar{\tau})^{i-1}P}=\frac{A_{i-1}}{Z_{(\mu\bar{\tau})^{i-2}P}}$, $X_{(\mu\bar{\tau})^{i-1}P}=A_{i-1}^2$ by Algorithm 5; when $i>3$, $x_{(\mu\bar{\tau})^{i-1}P}=\frac{A_{i-1}}{Z_{(\mu\bar{\tau})^{i-2}P}}$, $X_{(\mu\bar{\tau})^{i-1}P}=A_{i-1}^2$ by Algorithm 6. We have

$$\begin{aligned}
&(\mu\bar{\tau})^{i}P\\
=&\left(\frac{(\frac{A_{i-1}}{Z_{(\mu\bar{\tau})^{i-2}P}})^2+1}{\frac{A_{i-1}}{Z_{(\mu\bar{\tau})^{i-2}P}}},\frac{(\frac{A_{i-1}}{Z_{(\mu\bar{\tau})^{i-2}P}})^2}{(\frac{A_{i-1}}{Z_{(\mu\bar{\tau})^{i-2}P}})^2+1}+\frac{\Lambda_{(\mu\bar{\tau})^{i-1}P}}{Z_{(\mu\bar{\tau})^{i-1}P}}\right)\\
=&\left(\frac{A_{i-1}^2+Z_{(\mu\bar{\tau})^{i-2}P}^2}{A_{i-1}Z_{(\mu\bar{\tau})^{i-2}P}},\frac{A_{i-1}^2}{A_{i-1}^2+Z_{(\mu\bar{\tau})^{i-2}P}^2}+\frac{\Lambda_{(\mu\bar{\tau})^{i-1}P}}{Z_{(\mu\bar{\tau})^{i-1}P}}\right)\\
=&\left(\frac{X_{(\mu\bar{\tau})^{i-1}P}+Z_{(\mu\bar{\tau})^{i-2}P}^2}{Z_{(\mu\bar{\tau})^{i-1}P}},\right.\\
&\left.\frac{X_{(\mu\bar{\tau})^{i-1}P}}{X_{(\mu\bar{\tau})^{i-1}P}+Z_{(\mu\bar{\tau})^{i-2}P}^2}+\frac{\Lambda_{(\mu\bar{\tau})^{i-1}P}}{Z_{(\mu\bar{\tau})^{i-1}P}}\right).
\end{aligned}$$

Then $(\mu\bar{\tau})^{i}P$, $i\geq 3$ can be computed as

$$\begin{aligned}
A_i=&Z_{(\mu\bar{\tau})^{i-2}P}^2+X_{(\mu\bar{\tau})^{i-1}P},\\
X_{(\mu\bar{\tau})^{i}P}=&A_i^2,\\
\Lambda_{(\mu\bar{\tau})^{i}P}=&X_{(\mu\bar{\tau})^{i-1}P}Z_{(\mu\bar{\tau})^{i-1}P}+\Lambda_{(\mu\bar{\tau})^{i-1}P}A_i,\\
Z_{(\mu\bar{\tau})^{i}P}=&Z_{(\mu\bar{\tau})^{i-1}P}A_i.
\end{aligned}$$

It is described as Algorithm 6.

■

$\mu\bar{\tau}$-operation can be used to speed up scalar multiplications directly and used in double-base representation to improve the efficiency of scalar multiplications which are not the key concerns of this work. When $Z$-coordinate of $P$ is 1, by $\Lambda_{(\mu\bar{\tau})^2P}=X_{\mu\bar{\tau}P}Z_{\mu\bar{\tau}P}+\Lambda_{\mu\bar{\tau}P}A_2=A_2Z_{\mu\bar{\tau}P}\lambda_P+x_P$ in Algorithm 5 and Theorem 2, one directly gets the formulas of $\mu\bar{\tau}\&(\mu\bar{\tau})^2$-affine operation.

**Theorem 3** *Let $P=(x_P,\lambda_P)$. $\mu\bar{\tau}P$ and $(\mu\bar{\tau})^2P$ ($\mu\bar{\tau}\&(\mu\bar{\tau})^2$-affine operation) can be computed as Algorithm 7 at the cost of 4**M**+3**S**. The values of $\mu\bar{\tau}P$ and $(\mu\bar{\tau})^2P$ in Algorithm 7 are the same as those in Algorithm 4 with $Z_P=1$. Then the conditions of calculating $(\mu\bar{\tau})^{i}P$, $i\geq 3$ using Algorithm 7 are still satisfied.*

The costs of point operations including mixed addition, $(P\pm Q)$-operation, $\mu\bar{\tau}$-operation, re-$\mu\bar{\tau}$ operation, re-re-$\mu\bar{\tau}$ operation, and $\mu\bar{\tau}\&(\mu\bar{\tau})^2$-affine operation are summarized in Table 5. From Table 5, $\mu\bar{\tau}$-operation requires 5**M**+3**S**, re-$\mu\bar{\tau}$ operation and re-re-$\mu\bar{\tau}$ operation both require 3**M**+2**S**, $\mu\bar{\tau}\&(\mu\bar{\tau})^2$-affine operation requires 4**M**+3**S**. As a mixed point addition requires 8**M**+2**S**, these point operations are all very efficient.

**Algorithm 7** $\mu\bar{\tau}\&(\mu\bar{\tau})^2$-affine operation

**Input:** $P = (x_P, \lambda_P)$
**Output:** $\mu\bar{\tau}P = (X_{\mu\bar{\tau}P}, \Lambda_{\mu\bar{\tau}P}, Z_{\mu\bar{\tau}P})$, $(\mu\bar{\tau})^2P = (X_{(\mu\bar{\tau})^2P}, \Lambda_{(\mu\bar{\tau})^2P}, Z_{(\mu\bar{\tau})^2P})$
**Computation**

| | |
|---|---|
| 1) $\beta = x_P^2$ | **S** |
| 2) $X_{\mu\bar{\tau}P} = \beta^2 + 1$ | **S** |
| 3) $Z_{\mu\bar{\tau}P} = x_P\beta + x_P$ | **M** |
| 4) $\Lambda_{\mu\bar{\tau}P} = (\lambda_P + 1)Z_{\mu\bar{\tau}P} + x_P$ | **M** |
| 5) $A_2 = X_{\mu\bar{\tau}P} + \beta$ | |
| 6) $X_{(\mu\bar{\tau})^2P} = A_2^2$ | **S** |
| 7) $Z_{(\mu\bar{\tau})^2P} = A_2 Z_{\mu\bar{\tau}P}$ | **M** |
| 8) $\Lambda_{(\mu\bar{\tau})^2P} = Z_{(\mu\bar{\tau})^2P}\lambda_P + x_P$ | **M** |

**Table 5.** Costs of point operations

| Point operation | cost | algorithm |
|---|---|---|
| mixed addition | 8**M**+2**S** | |
| $(P \pm Q)$-operation (this work) | 12**M**+5**S** | Algorithm 3 |
| $\mu\bar{\tau}$-operation (this work) | 5**M**+3**S** | Algorithm 4 |
| re-$\mu\bar{\tau}$ operation (this work) | 3**M**+2**S** | Algorithm 5 |
| re-re-$\mu\bar{\tau}$ operation (this work) | 3**M**+3**S** | Algorithm 6 |
| $\mu\bar{\tau}\&(\mu\bar{\tau})^2$-affine operation (this work) | 4**M**+3**S** | Algorithm 7 |

Based on the efficient formulas of $(P \pm Q)$-operation, $\mu\bar{\tau}$-operation, re-$\mu\bar{\tau}$ operation, re-re-$\mu\bar{\tau}$ operation, and $\mu\bar{\tau}\&(\mu\bar{\tau})^2$-affine operation, we will introduce a novel scheme to improve the efficiency of pre-computation stage of window $\tau$NAF on Koblitz curves.

## 4 A Novel Pre-Computation Scheme

Solinas' pre-computation; Hankerson, Menezes, and Vanstone's pre-computation; and Trost and Xu's pre-computation all have a pre-computation scheme on $E_0$ and another pre-computation scheme on $E_1$. For convenience, we define a unified pre-computation.

**Definition 1 (Unified pre-computation)** *If a pre-computation scheme computing all $Q_i$ for $i \in I_w$ works on both $E_0$ and $E_1$, we call it a unified pre-computation.*

To design an efficient unified pre-computation, some properties of $R_i$, $i \in I_w$ are demanded.

### 4.1 Basic Lemmas

Recall that for $w \geq 3$, $I_w = \{1, 3, \cdots, 2^{w-1} - 1\}$ and $R_i$ consists of the elements of the class $i$ modulo $\tau^w$ whose norms are smaller than $2^w$, for each $i \in I_w$. Since elements of $I_w$ are odd integers, we will mainly work on the subset $(2\mathbb{Z}+1)+\mathbb{Z}\tau \subset \mathbb{Z}[\tau]$ as $R_i \subset (2\mathbb{Z} + 1) + \mathbb{Z}\tau$.

**Lemma 2** *We have the following facts:*

1. *If $g + h\tau \in R_i$ for some $i \in I_w$, then $g - h\tau \notin R_i$.*
2. *If $g + h\tau \in R_i$ for some $i \in I_w$, then $g' + h\tau \notin R_i$ for any $g' \in \mathbb{Z} \setminus \{g\}$.*
3. *For any $g + h\tau \in (2\mathbb{Z} + 1) + \mathbb{Z}\tau$, there exists an $i \in I_w$ such that $i \equiv g + h\tau \pmod{\tau^w}$ or $-i \equiv g + h\tau \pmod{\tau^w}$.*

*Proof.* From [6], we know that if $g + h\tau \in R_i$, then $|g| < \frac{2^{\frac{w+2}{2}}}{\sqrt{3}}$, $|h| < 2^{\frac{w}{2}}$ where $|g|$ denotes its absolute value.

(1) Assume both $g+h\tau$ and $g-h\tau$ are in $R_i$, then $\tau^w | 2h\tau$. By Equation (1), this implies that $2^w | 2hs_w$ and hence $2^{w-2} | h$ as $\frac{s_w}{2}$ is odd. On the other hand, since $N(g \pm h\tau) < 2^w$, we see that $h^2 < 2^{w-1}$. This reaches a contradiction.

(2) Assume both $g+h\tau$ and $g'+h\tau$ are in $R_i$ for some $g' \neq g$, then $\tau^w | (g-g')$. We get $2^w | (g - g')$ by Equation (1). Since $|g|, |g'| < \frac{2^{\frac{w+2}{2}}}{\sqrt{3}}$, we get $|g - g'| < 2\frac{2^{\frac{w+2}{2}}}{\sqrt{3}} \leq 2^w$. We get a contradiction again.

(3) Since $g + hs_w$ is odd, it must be in one of the congruence classes of $-2^{w-1} + 1, -2^{w-1} + 3, \ldots, -3, -1, , 1, 3, \ldots, 2^{w-1} - 3, 2^{w-1} - 1$ modulo $2^w$. ∎

We can show that the number of elements of $R_i$ is well bounded.

**Lemma 3** *Let $i \in I_w$, then*

$$|R_i| \leq \lfloor 2^{\frac{w+2}{2}} \rfloor.$$

*Proof.* If $g + h\tau \in R_i$, we know that $|h| < 2^{\frac{w}{2}}$. So $T = \{h \in \mathbb{Z} | g + h\tau \in R_i \text{ for some odd number } g\}$ has less than $2 \cdot 2^{\frac{w}{2}}$ number of elements. By Lemma 2, for each $h \in T$, there is only one $g$ available such that $g + h\tau \in R_i$. Thus

$$|R_i| = |T| \leq \lfloor 2^{\frac{w+2}{2}} \rfloor.$$

∎

If $g + h_1\tau \equiv g + h_2\tau \pmod{\tau^w}$, $s_w(h_2 - h_1) \equiv 0 \pmod{2^w}$. $s_w$ is even, and $s_w/2$ is odd, then $h_2 = h_1 + c \cdot 2^{w-1}$. Thus $g+h\tau, g+(h+1)\tau, \ldots, g+(h+2^{w-1}-1)\tau$ cover all congruence classes $R_i$ and $R_{-i}$, $i \in I_w$ where $g$ is odd. For each $i$, there is $\frac{\lfloor \frac{2^{\frac{w+2}{2}}}{\sqrt{3}} \rfloor \cdot \lfloor 2^{\frac{w}{2}} \rfloor}{2^{w-2}} < 4.62$ solutions of $g + h\tau$ satisfying $i \equiv g + h\tau \pmod{\tau^w}$ on average. We have calculated out that $|R_i| \leqslant 3$ for $i \in I_w$ when $3 \leq w \leq 10$.

### 4.2 A Unified Pre-Computation

Let $c_i \in R_i$ and $c_i = g + h\mu\tau$. Then $N(g + h\mu\tau) = g^2 + gh + 2h^2 < 2^w$. This fact that the norm of $g + h\mu\tau$ has no concern with $\mu$ is the premise of our unified pre-computation. We propose a plane search to generate $R_i$, $i \in I_w$ shown as Algorithm 8. For each $(g + h\mu\tau) \in (2\mathbb{Z} + 1) + \mathbb{Z}\tau$ with $N(g + h\mu\tau) = g^2+gh+2h^2 < 2^w$, we dot it on a plane with $g$ as horizontal axis and $h$ as vertical axis. To determine whether $g+h\mu\tau$ is in $R_i$ for some $i$ satisfying $2^w | g-i+h\mu s_w$,

we search all points $(g, h)$ where $-\lfloor 2^{\frac{w+2}{\sqrt{3}}} \rfloor \leq g \leq \lfloor 2^{\frac{w+2}{\sqrt{3}}} \rfloor$, $-\lfloor 2^{\frac{w}{2}} \rfloor \leq h \leq \lfloor 2^{\frac{w}{2}} \rfloor$ and $g$ is odd. We call this method plane search. For $s_w$ has a unified representation and by Algorithm 2, $\mu s_w$ is an integer associated with $w$ but no connection with $\mu$ or $a$. Then one can figure out each element of $R_i$ associated with $u$ by Algorithm 8. $Q_i = c_i P$ works on both $E_0$ and $E_1$ for $c_i = g + h\mu\tau \in R_i$.

**Algorithm 8** Plane search to generate $R_i$, $i \in I_w$

**Computation**

1. $R_i \leftarrow <>$
2. for $h \leftarrow -\lfloor 2^{\frac{w}{2}} \rfloor$ to $\lfloor 2^{\frac{w}{2}} \rfloor$
   for $g \leftarrow -\lfloor 2^{\frac{w+2}{\sqrt{3}}} \rfloor$ to $\lfloor 2^{\frac{w+2}{\sqrt{3}}} \rfloor$ and $g$ is odd
   if$(2^w | g - i + h\mu s_w)$ and $(g^2 + gh + 2h^2 < 2^w)$
   then append $(g + h\mu\tau)$ to $R_i$
3. output $R_i$

We collect all such elements and form a set

$$C = \{c_i | i \in I_w\}.$$

Then $Q_i = c_i P, c_i \in C$ suits for both $E_0$ and $E_1$. These $Q_i = c_i P, c_i \in C, i \in I_w$ form a unified pre-computation. We always set the trivial case $c_1 = 1$. Then $C$ can be briefly denoted as $C = \{c_i | i \in I_w, i \neq 1\}$.

### 4.3 Our Novel Pre-Computation

At present, we design a novel pre-computation for window $\tau$NAF with width 4, 5, and 6.

**Theorem 4** *Let $P = (x_P, \lambda_P)$, $Q_i = (X_i, \Lambda_i, Z_i)$, $i \in I_w$. There exist unified pre-computation schemes shown as Algorithm 9, Algorithm 10, and Algorithm 11 requiring* $7\mathbf{M}+5\mathbf{S}$*,* $26\mathbf{M}+16\mathbf{S}$*, and* $66\mathbf{M}+36\mathbf{S}$ *for window $\tau$NAF with width $w = 4$, 5, and 6 respectively. In Algorithm 9, "$(-Q_5, -Q_7)$=$\mu\bar{\tau}$&$(\mu\bar{\tau})^2$-affine(P) by Algorithm 7" means that $(-Q_5, -Q_7)$ is the output of Algorithm 7 with input $P$. The same representation suits for other sentences in Algorithms 9, 10, and 11. The explicit design is shown as Table 6.*

*Proof.* An algorithm for calculating pre-computations for window $\tau$NAF with width 4, 5, and 6 is shown in Table 6. Let $c_i = g + h\mu\tau$ for each $i \in I_w$ in Table 6. Since $c_i = g + h\mu\tau$ for each $i \in I_w$ with $w = 4$, 5, and 6, the pre-computation in Table 6 is a unified pre-computation. It is obvious to verify that $g + h\mu s_w \equiv i \pmod{2^w}$ and $N(c_i) < 2^w$ for each $i \in I_w$. This ensures the correctness of our unified pre-computation.

At present, we count the field operations of our unified pre-computation.

The calculation process is as follows.

**Algorithm 9** Pre-computation for $w = 4$

**Input:** $P = (x_P, \lambda_P)$
**Output:** $Q_i$, $i \in I_w$
**Computation**

1. $(-Q_5, -Q_7)$=$\mu\bar{\tau}\&(\mu\bar{\tau})^2$-affine$(P)$ by Algorithm 7
2. $Q_3$=re-re-$\mu\bar{\tau}(-Q_7, Z_5)$ by Algorithm 6

**Algorithm 10** Pre-computation for $w = 5$

**Input:** $P = (x_P, \lambda_P)$
**Output:** $Q_i$, $i \in I_w$
**Computation**

1. $\mu\tau P = (x_P^2, \lambda_P^2 + a + 1)$
2. $(-Q_5, -Q_7)$=$\mu\bar{\tau}\&(\mu\bar{\tau})^2$-affine$(P)$ by Algorithm 7
3. $Q_3$=re-re-$\mu\bar{\tau}(-Q_7, Z_5)$ by Algorithm 6
4. $-Q_{15}$=re-re-$\mu\bar{\tau}(Q_3, Z_7)$ by Algorithm 6
5. $Q_{11} = \mu\tau P + Q_5$
6. $(Q_9, \alpha) = \mu\bar{\tau}(Q_{11})$ by Algorithm 4
7. $Q_3$=re-$\mu\bar{\tau}(Q_9, \alpha)$ by Algorithm 5

**Algorithm 11** Pre-computation for $w = 6$

**Input:** $P = (x_P, \lambda_P)$
**Output:** $Q_i$, $i \in I_w$
**Computation**

1. $\mu\tau P = (x_P^2, \lambda_P^2 + a + 1)$
2. $(Q_{27}, Q_{25})$=$\mu\bar{\tau}\&(\mu\bar{\tau})^2$-affine$(P)$ by Algorithm 7
3. $-Q_{29}$=re-re-$\mu\bar{\tau}(Q_{25}, Z_{27})$ by Algorithm 6
4. $Q_{15}$=re-re-$\mu\bar{\tau}(Q_{29}, Z_{25})$ by Algorithm 6
5. $Q_{21}$=re-re-$\mu\bar{\tau}(Q_{15}, Z_{29})$ by Algorithm 6
6. $(Q_9, Q_3)$=$(P \pm Q)$-operation$(\mu\tau P, Q_{29})$ by Algorithm 3
7. $(-Q_{13}, \alpha) = \mu\bar{\tau}(Q_9)$ by Algorithm 4
8. $Q_{31}$=re-$\mu\bar{\tau}(Q_{13}, \alpha)$ by Algorithm 5
9. $(Q_{17}, \alpha) = \mu\bar{\tau}(Q_{11})$ by Algorithm 4
10. $Q_{11}$=re-$\mu\bar{\tau}(Q_{17}, \alpha)$ by Algorithm 5
11. $Q_{23} = -Q_{15} + \mu\tau P$.
12. $(-Q_{19}, \alpha) = \mu\bar{\tau}(Q_{11})$ by Algorithm 4
13. $Q_5$, $Q_7$=$(P \pm Q)$-operation$(\mu\tau P, Q_{31})$ by Algorithm 3

**Table 6.** Novel pre-computation for $w = 4, 5, 6$

| | | | |
|---|---|---|---|
| w=4 | | | 7**M**+5**S** |
| $c_5 = -1 + \mu\tau$ | | $Q_5 = -\mu\bar{\tau}P$ | 2**M**+2**S** |
| $c_7 = 1 + \mu\tau$ | $c_7 = \mu\bar{\tau}c_5$ | $Q_7 = -(\mu\bar{\tau})^2P$ | 2**M**+**S** |
| $c_3 = -3 + \mu\tau$ | $c_3 = -\mu\bar{\tau}c_7$ | $Q_3 = (\mu\bar{\tau})^3P$ | 3**M**+2**S** |
| | | | |
| w=5 | | | 26**M**+16**S** |
| $c_5 = -1 + \mu\tau$ | | $Q_5 = -\mu\bar{\tau}P$ | 2**M**+2**S** |
| $c_7 = 1 + \mu\tau$ | $c_7 = \mu\bar{\tau}c_5$ | $Q_7 = -(\mu\bar{\tau})^2P$ | 2**M**+**S** |
| $c_3 = -3 + \mu\tau$ | $c_3 = -\mu\bar{\tau}c_7$ | $Q_3 = (\mu\bar{\tau})^3P$ | 3**M**+2**S** |
| $c_{15} = 1 - 3\mu\tau$ | $c_{15} = -\mu\bar{\tau}c_3$ | $Q_{15} = -(\mu\bar{\tau})^4P$ | 3**M**+2**S** |
| $c_{11} = -1 + 2\mu\tau$ | $c_{11} = \mu\tau + c_5$ | $Q_{11} = \mu\tau P + Q_5$ | 8**M**+4**S** |
| $c_9 = 3 + \mu\tau$ | $c_9 = \mu\bar{\tau}c_{11}$ | $Q_9 = \mu\bar{\tau}Q_{11}$ | 5**M**+3**S** |
| $c_{13} = -5 + 3\mu\tau$ | $c_{13} = -\mu\bar{\tau}c_9$ | $Q_{13} = -(\mu\bar{\tau})^2Q_{11}$ | 3**M**+2**S** |
| | | | |
| w=6 | | | 66**M**+36**S** |
| $c_{27} = 1 - \mu\tau$ | | $Q_{27} = \mu\bar{\tau}P$ | 2**M**+2**S** |
| $c_{25} = -1 - \mu\tau$ | $c_{25} = \mu\bar{\tau}c_{27}$ | $Q_{25} = (\mu\bar{\tau})^2P$ | 2**M**+**S** |
| $c_{29} = 3 - \mu\tau$ | $c_{29} = -\mu\bar{\tau}c_{25}$ | $Q_{29} = -(\mu\bar{\tau})^3P$ | 3**M**+2**S** |
| $c_{15} = 1 - 3\mu\tau$ | $c_{15} = \mu\bar{\tau}c_{29}$ | $Q_{15} = -(\mu\bar{\tau})^4P$ | 3**M**+2**S** |
| $c_{21} = -5 - \mu\tau$ | $c_{21} = \mu\bar{\tau}c_{15}$ | $Q_{21} = -(\mu\bar{\tau})^5P$ | 3**M**+2**S** |
| $c_3 = 3$ | $c_3 = \mu\tau + c_{29}$ | $Q_3 = \mu\tau P + Q_{29}$ | |
| $c_9 = -3 + 2\mu\tau$ | $c_9 = \mu\tau - c_{29}$ | $Q_9 = \mu\tau P - Q_{29}$ | 12**M**+7**S** |
| $c_{13} = -1 - 3\mu\tau$ | $c_{13} = -\mu\bar{\tau}c_9$ | $Q_{13} = -(\mu\bar{\tau})Q_9$ | 5**M**+3**S** |
| $c_{31} = -7 + \mu\tau$ | $c_{31} = \mu\bar{\tau}c_{13}$ | $Q_{31} = -(\mu\bar{\tau})^2Q_9$ | 3**M**+2**S** |
| $c_{17} = 3 - 3\mu\tau$ | $c_{17} = \mu\bar{\tau}c_3$ | $Q_{17} = \mu\bar{\tau}Q_3$ | 5**M**+3**S** |
| $c_{11} = -3 - 3\mu\tau$ | $c_{11} = \mu\bar{\tau}c_{17}$ | $Q_{11} = (\mu\bar{\tau})^2Q_3$ | 3**M**+2**S** |
| $c_{23} = -1 + 4\mu\tau$ | $c_{23} = \mu\tau - c_{15}$ | $Q_{23} = \mu\tau P - Q_{15}$ | 8**M**+2**S** |
| $c_{19} = -7 - \mu\tau$ | $c_{19} = -\mu\bar{\tau}c_{23}$ | $Q_{19} = -\mu\bar{\tau}Q_{23}$ | 5**M**+3**S** |
| $c_5 = -7 + 2\mu\tau$ | $c_5 = \mu\tau + c_{31}$ | $Q_5 = \mu\tau P + Q_{31}$ | |
| $c_7 = 7$ | $c_7 = \mu\tau - c_{31}$ | $Q_7 = \mu\tau P - Q_{31}$ | 12**M**+5**S** |

1. $w = 4$. $Q_5 = -(\mu\bar{\tau}P)$, $Q_7 = -(\mu\bar{\tau})^2P$, $Q_3 = (\mu\bar{\tau})^3P$ are shown as Algorithm 9 and Table 6. It requires 1 $\mu\bar{\tau}$&$(\mu\bar{\tau})^2$-affine operation and 1 re-re-$\mu\bar{\tau}$ operation which costs 7**M**+5**S**.
2. $w = 5$. Let $\tau P = (x_{\tau P}, \lambda_{\tau P}) = (x_P^2, \lambda_P^2)$. $Q_5 = -(\mu\bar{\tau}P)$, $Q_7 = -(\mu\bar{\tau})^2P$, $Q_3 = (\mu\bar{\tau})^3P$, $Q_{15} = -(\mu\bar{\tau})^4P$, $Q_{11} = \mu\tau P + Q_5$, $Q_9 = \mu\bar{\tau}Q_{11}$, $Q_{13} = -(\mu\bar{\tau})^2Q_{11}$ are shown as Algorithm 10 and Table 6. It requires 1 $\mu\bar{\tau}$&$(\mu\bar{\tau})^2$-affine operation, 1 $\mu\bar{\tau}$-operation, 2 re-$\mu\bar{\tau}$ operations, 1 re-re-$\mu\bar{\tau}$ operation, 1 $\tau$-affine operation, and 1 mixed addition which costs 26**M**+16**S**.
3. $w = 6$. Let $\tau P = (x_{\tau P}, \lambda_{\tau P}) = (x_P^2, \lambda_P^2)$.
   $Q_{27} = \mu\bar{\tau}P$, $Q_{25} = (\mu\bar{\tau})^2P$, $Q_{29} = -(\mu\bar{\tau})^3P$, $Q_{15} = -(\mu\bar{\tau})^4P$, $Q_{21} = -(\mu\bar{\tau})^5P$, $(Q_3, Q_9) = \mu\tau P \pm Q_{29}$ ($Q_3 = \mu\tau P + Q_{29}$, $Q_9 = \mu\tau P - Q_{29}$), $Q_{13} = -(\mu\bar{\tau})Q_9$, $Q_{31} = -(\mu\bar{\tau})^2Q_9$, $Q_{17} = \mu\bar{\tau}Q_3$, $Q_{11} = (\mu\bar{\tau})^2Q_3$, $Q_{23} = \mu\tau P - Q_{15}$, $Q_{19} = -\mu\bar{\tau}Q_{23}$, $(Q_5, Q_7) = \mu\tau P \pm Q_{31}$ ($Q_5 = \mu\tau P + Q_{31}$, $Q_7 = \mu\tau P - Q_{31}$) are shown as Algorithm 11 and Table 6.
   It requires 1 $\mu\bar{\tau}$&$(\mu\bar{\tau})^2$-affine operation, 3 $\mu\bar{\tau}$-operations, 2 re-$\mu\bar{\tau}$ operations, 3 re-re-$\mu\bar{\tau}$ operations, 1 $\tau$-affine operation, 1 mixed addition, and 2 $(P \pm Q)$-operations which costs 66**M**+36**S**.

The explicit computing process and the value of $c_i$ are shown as Table 6. ■

The design of this unified pre-computation is shown in the Appendix A. As shown in [6], for each $Q_i$ ($i = 3, 5, \ldots, 2^{w-1} - 1$), one point addition is necessary.

In this work, we employ more efficient $\mu\bar{\tau}$-operations to replace point addition. These operations contain $\mu\bar{\tau}$-operation, re-$\mu\bar{\tau}$ operation, re-re-$\mu\bar{\tau}$ operation, and $\mu\bar{\tau}\&(\mu\bar{\tau})^2$-affine operation which lead to the speedup of our pre-computation algorithm. Next, we will compare these pre-computation schemes.

### 4.4 Comparison of Pre-Computation Schemes in M and S

It is standard to compare these schemes by counting field multiplications and field squarings. Suppose that $\mathbf{I}/\mathbf{M}$ =10 and $\mathbf{S}/\mathbf{M}$ =0 or 0.2 which is suggested by Bernstein and Lange in their explicit formulas database [14]. These ratios are reasonable in the experiments of our environments shown as Section 6 where $\mathbf{I}/\mathbf{M}$ =10 and 0< $\mathbf{S}/\mathbf{M}$< 0.2. The costs of Solinas' pre-computation scheme; Hankerson, Menezes, and Vanstone's pre-computation scheme; Trost and Xu's pre-computation scheme; and our pre-computation scheme are summarized in Table 1. From Table 1, our pre-computation scheme is the fastest one among these 4 pre-computation schemes.

From Table 1, our novel pre-computation scheme is about 110% faster than Solinas' pre-computation scheme for $w = 4$ and 70% for $w = 5$ in both cases of $\mathbf{S}$=0$\mathbf{M}$ and $\mathbf{S}$=0.2$\mathbf{M}$. It is about 110% faster than Hankerson, Menezes, and Vanstone's pre-computation scheme for $w = 4$, 90% for $w = 5$, and 77% for $w = 6$. Our unified pre-computation is about 60% faster than Trost and Xu's pre-computation scheme for $w = 4$, 5, and 6.

## 5 Scalar Multiplications

Let the costs of pre-computation schemes for window $\tau$NAF with width $w$ in Table 1 be denoted by $Pre_w$. Scalar multiplication using window $\tau$NAF has two situations.

1. Scalar multiplication uses pre-computations in $\lambda$-projective coordinates. It requires $m$ $\tau$-operations, $\frac{m}{w+1}\frac{2^{w-2}-1}{2^{w-2}}$ point additions, $\frac{m}{w+1}\frac{1}{2^{w-2}}$ mixed additions, and the pre-computation. Scalar multiplication in this case costs
$$3m\mathbf{S} + \frac{m}{w+1}(11\mathbf{M} + 2\mathbf{S} - \frac{3}{2^{w-2}}\mathbf{M}) + Pre_w. \quad (3)$$
2. Scalar multiplication uses pre-computations in $\lambda$-coordinates. This method can fully use mixed additions and requires Montgomery trick to translate the pre-computation points in $\lambda$-coordinates to points in $\lambda$-coordinates. It requires $m$ $\tau$-projective operations, $\frac{m}{w+1}$ mixed additions, Montgomery trick, and the pre-computation. Scalar multiplication in this case costs
$$3m\mathbf{S} + \frac{m}{w+1}(8\mathbf{M} + 2\mathbf{S}) + \mathbf{I} + (5 \cdot 2^{w-2} - 8)\mathbf{M} + Pre_w. \quad (4)$$

The costs of scalar multiplications for both cases are shown in Table 7.

For $w = 4$, we choose Case 1 to compute scalar multiplications when $I \geq (\frac{9m}{20} - 12)\mathbf{M}$, i.e., $m \leq 48$. For $w = 5$, we choose Case 1 to compute scalar

**Table 7.** The costs of scalar multiplications using window $\tau$NAF

| case | case 1 | case 2 |
|---|---|---|
| $w=4$ | $\frac{41m}{20}\mathbf{M}+\frac{17m}{5}\mathbf{S}+Pre_4$ | $\mathbf{I}+(\frac{8m}{5}+12)\mathbf{M}+\frac{17m}{5}\mathbf{S}+Pre_4$ |
| $w=5$ | $\frac{85m}{48}\mathbf{M}+\frac{10m}{3}\mathbf{S}+Pre_5$ | $\mathbf{I}+(\frac{4m}{3}+32)\mathbf{M}+\frac{10m}{3}\mathbf{S}+Pre_5$ |
| $w=6$ | $\frac{173m}{112}\mathbf{M}+\frac{23m}{7}\mathbf{S}+Pre_6$ | $\mathbf{I}+(\frac{8m}{7}+72)\mathbf{M}+\frac{23m}{7}\mathbf{S}+Pre_6$ |

multiplications when $I \geq (\frac{7m}{17}-32)\mathbf{M}$, i.e., $m \leq 102$. For $w=6$, we choose Case 1 to compute scalar multiplications when $I \geq (\frac{45m}{112}-72)\mathbf{M}$, i.e., $m \leq 204$. It is obvious that we should perform scalar multiplications on K-163, K-233, K-283, K-409, and K-571 using Case 2 to get a higher performance.

We summarized the lowest costs of scalar multiplications on each curve using our pre-computation scheme and the costs of scalar multiplications using Trost and Xu's pre-computation scheme by Equations (3) and (4) in Table 8. On K-163, scalar multiplications using our pre-computation scheme utilize $w=5$ and those using Trost and Xu's scheme utilize $w=4$. On K-233, K-283, and K-409, $w=5$ is the best choice. On K-571, $w=6$ is the best choice. When $\mathbf{S}/\mathbf{M}$ =0, scalar multiplications using our scheme are 3.0%, 4.5%, 3.8%, 2.8%, and 5.1% faster than those using Trost and Xu's scheme on K-163, K-233, K-283, K-409, and K-571 respectively. When $\mathbf{S}/\mathbf{M}$ =0.2, the ratios are 2.3%, 3.2%, 2.7%, 2.0%, and 3.5% respectively.

**Table 8.** The costs of scalar multiplications on K-163, K-233, K-283, K-409, and K-571 in $\mathbf{M}$

| Curve | K-163($w$) | K-233($w$) | K-283($w$) | K-409($w$) | K-571($w$) |
|---|---|---|---|---|---|
| $\mathbf{S}$=0$\mathbf{M}$ | | | | | |
| Trost, Xu | 293.8(4) | 395.7(5) | 462.3(5) | 630.3(5) | 841.5(6) |
| our | 285.3(5) | 378.7(5) | 445.3(5) | 613.3(5) | 800.5(6) |
| $\mathbf{S}$=0.2$\mathbf{M}$ | | | | | |
| Trost, Xu | 406.2(4) | 554.6(5) | 654.6(5) | 906.6(5) | 1224(6) |
| our | 397.2(5) | 537.2(5) | 637.2(5) | 889.2(5) | 1183(6) |

## 6 Experiments

Miracl lib [15] is used to implement field arithmetics over $\mathbb{F}_{2^m}$. Our experiments are tested by using C++ programs which are compiled by Microsoft visual studio 2008 on an Intel Core 2. The processor speed is 2.66 GHz processor and the operating system is 32-bit Windows XP. This section will present some experimental results of different pre-computation schemes, scalar multiplications using our scheme and those using scheme in [6].

### 6.1 Pre-Computation Schemes

We ran each pre-computation scheme 1000 times on five Koblitz curves including K-163, K-233, K-283, K-409, and K-571. The time costs of our experiments for $w$ from 4 to 6 are shown in Table 9.

On K-163, K-233, K-283, K-409, and K-571, Table 9 shows that our pre-computation scheme is about 110%, 110%, 60% faster than Solinas' scheme;

**Table 9.** Time cost of pre-computations on K-163, K-233, K-283, K-409, and K-571 in $\mu$s

| Curve | K-163 | K-233 | K-283 | K-409 | K-571 |
|---|---|---|---|---|---|
| $w = 4$ | | | | | |
| Solinas | 80 | 101 | 119 | 210 | 295 |
| Hankerson, Menezes, Vanstone | 80 | 101 | 119 | 210 | 295 |
| Trost, Xu | 63 | 80 | 92 | 161 | 224 |
| Ours | 39 | 49 | 57 | 100 | 141 |
| $w = 5$ | | | | | |
| Solinas | 210 | 270 | 328 | 591 | 845 |
| Hankerson, Menezes, Vanstone | 228 | 298 | 358 | 659 | 950 |
| Trost, Xu | 203 | 271 | 304 | 554 | 810 |
| Ours | 125 | 163 | 189 | 341 | 502 |
| $w = 6$ | | | | | |
| Hankerson, Menezes, Vanstone | 542 | 709 | 846 | 1502 | 2179 |
| Trost, Xu | 473 | 617 | 737 | 1323 | 1977 |
| Ours | 303 | 393 | 458 | 820 | 1176 |

Hankerson, Menezes, and Vanstone's scheme; and Trost and Xu's scheme respectively for $w = 4$; about 70%, 90%, 60% faster than these three pre-computation schemes respectively for $w = 5$; about 80%, 60% faster than Hankerson, Menezes, and Vanstone's scheme; and Trost and Xu's pre-computation scheme respectively for $w = 6$ .

Within the bounds of the error, the practical implementations are consistent with the theoretical analysis. The reason of some differences is that some field additions are ignorant and that the ratio of **M** and **S** is different.

### 6.2 Scalar multiplications

The experimental results of scalar multiplications using window $\tau$NAF on the standardized curves NIST K-163, K-233, K-283, K-409, and K-571 are shown in Table 10. On K-163, scalar multiplications using our pre-computation scheme are 2.8% faster than those using Trost and Xu's pre-computation scheme. On K-233, K-283, and K-409, scalar multiplications using our scheme are 4.2%, 3.1%, and 2.5% faster than those using Trost and Xu's scheme respectively. Scalar multiplications use window $\tau$NAF with $w = 5$ on K-233, K-283, and K-409. Since our pre-computation scheme saves 18**M**+2**S**, the cost of scalar multiplications increase as $m$ increases and the ratio of improvement is reduced. For example, the improvement of scalar multiplication on K-233 is larger than that on K-283. On K-571, scalar multiplications using our pre-computation scheme are 4.9% faster than those using Trost and Xu's pre-computation scheme. Both scalar multiplications use window $\tau$NAF with $w = 6$ on K-571. In this case, our pre-computation scheme saves 42**M** which is more than 18**M**+2**S** in the case of $w = 5$. This is the reason that the ratio of improvement of scalar multiplication is 4.9% on K-571, and 4.2%, 3.1%, 2.5% on K-233, K-283, and K-409. These experimental results confirm that our improvement of pre-computation can actually speed up the efficiency of scalar multiplications.

**Table 10.** Time cost of scalar multiplications in $\mu$s

| Curve | K-163 | K-233 | K-283 | K-409 | K-571 |
|---|---|---|---|---|---|
| Trost, Xu | 1559 | 2686 | 3739 | 9175 | 16889 |
| our | 1516 | 2578 | 3625 | 8953 | 16103 |

## 7 Conclusion

We introduce the notion of $\mu\bar{\tau}$-operation and design a novel pre-computation scheme on Koblitz curves. Results show that our pre-computation scheme is the most efficient one among all existing schemes. Scalar multiplications using our pre-computation scheme is about from 2.5% to 4.9% faster than those using Trost and Xu's scheme on the five Koblitz curves.

## A The Design of Our Novel Pre-Computation Scheme

Let $Q = (g + h\mu\tau)P$. $\mu\bar{\tau}Q = (g + 2h - \mu g\tau)P$. By Algorithm 8, we have $|g| \leq 7, |h| \leq 6$ for the cases of $w = 4$, 5, and 6. All possible cases of $\mu\bar{\tau}Q$ for $w = 4$, 5, and 6 are shown in Table 11.

**Table 11.** The values of $\mu\bar{\tau}Q$

| $Q$ | $\mu\bar{\tau}Q$ |
|---|---|
| $Q = P$ | $(1 - \mu\tau)P$ |
| $Q = (1 - \mu\tau)P$ | $(-1 - \mu\tau)P$ |
| $Q = (1 + \mu\tau)P$ | $(3 - \mu\tau)P$ |
| $Q = (1 + 2\mu\tau)P$ | $(5 - \mu\tau)P$ |
| $Q = (1 - 2\mu\tau)P$ | $(-3 - \mu\tau)P$ |
| $Q = (1 + 3\mu\tau)P$ | $(7 - \mu\tau)P$ |
| $Q = (1 - 3\mu\tau)P$ | $(-5 - \mu\tau)P$ |
| $Q = (1 - 4\mu\tau)P$ | $(-7 - \mu\tau)P$ |
| $Q = 3P$ | $(3 - 3\mu\tau)P$ |
| $Q = (3 + \mu\tau)P$ | $(5 - 3\mu\tau)P$ |
| $Q = (3 - \mu\tau)P$ | $(1 - 3\mu\tau)P$ |
| $Q = (3 + 2\mu\tau)P$ | $(7 - 3\mu\tau)P$ |
| $Q = (3 - 2\mu\tau)P$ | $(-1 - 3\mu\tau)P$ |
| $Q = (3 - 3\mu\tau)P$ | $(-3 - 3\mu\tau)P$ |
| $Q = (3 - 4\mu\tau)P$ | $(-5 - 3\mu\tau)P$ |
| $Q = (3 - 5\mu\tau)P$ | $(-7 - 3\mu\tau)P$ |
| $Q = 5P$ | $(5 - 5\mu\tau)P$ |
| $Q = (5 + \mu\tau)P$ | $(7 - 5\mu\tau)P$ |
| $Q = (5 - \mu\tau)P$ | $(3 - 5\mu\tau)P$ |
| $Q = (5 - 2\mu\tau)P$ | $(1 - 5\mu\tau)P$ |
| $Q = (5 - 3\mu\tau)P$ | $(-1 - 5\mu\tau)P$ |
| $Q = (5 - 4\mu\tau)P$ | $(-3 - 5\mu\tau)P$ |
| $Q = (5 - 5\mu\tau)P$ | $(-5 - 5\mu\tau)P$ |
| $Q = (5 - 6\mu\tau)P$ | $(-7 - 5\mu\tau)P$ |

The key technique is to choose suitable $c_i \in R_i$ to get a high efficiency of pre-computation scheme.

1. For the case of $w = 4$, by our plane search of Algorithm 8, $R_1 = \{1\}$, $R_3 = \{-1 - 2\mu\tau, 3, -3 + \mu\tau\}$, $R_5 = \{-1 + \mu\tau,\ 1 - 2\mu\tau\}$, $R_7 = \{1 + \mu\tau,\ -3 - \mu\tau,\ 3 - 2\mu\tau\}$.

For the trivial case $c_1$=1, $Q_1 = c_1P = P$. $\mu\bar{\tau}\&(\mu\bar{\tau})^2$-affine operation only requires 4**M**+3**S**. By Lemma 2, $\mu\bar{\tau} \in R_i$ or $-\mu\bar{\tau} \in R_i$ for some $i \in I_w$. The same for $(\mu\bar{\tau})^2$ where $(\mu\bar{\tau})^2 \in R_j$ or $-(\mu\bar{\tau})^2 \in R_j$ for some $j \in I_w$. Without loss of generality, suppose that $\mu\bar{\tau} \in R_i$ and $(\mu\bar{\tau})^2 \in R_j$. If any one or two of $\mu\bar{\tau}P$=$c_iP$, $(\mu\bar{\tau})^2P$=$c_jP$ is calculated as others ($c_i, c_j$ are other elements in $R_i, R_j$), the total cost will be more. Then $c_i = \mu\bar{\tau}$ and $c_j = (\mu\bar{\tau})^2$ is first determined for the nontrivial case. We first set $c_5 = -1 + \mu\tau = -\mu\bar{\tau}$, $c_7 = 1 + \mu\tau = -(\mu\bar{\tau})^2$.

If $c_3 = -1 - 2\mu\tau$, computing $Q_3$ requires 8**M**+4**S**; if $c_3 = 3$, more than 8**M**+4**S**; if $c_3 = -3 + \mu\tau = (\mu\bar{\tau})^3$, 3**M**+2**S**. Thus we choose that $C = \{c_5 = -1 + \mu\tau,\ c_7 = 1 + \mu\tau,\ c_3 = -3 + \mu\tau\}$ shown as Table 6.

2. For the case of $w = 5$, $R_1 = \{1,\ -5 + \mu\tau\}$, $R_3 = \{3,\ -3 + \mu\tau\}$, $R_5 = \{-1 + \mu\tau,\ 5\}$, $R_7 = \{1 + \mu\tau,\ -5 + 2\mu\tau\}$, $R_9 = \{1 - 4\mu\tau,\ 3 + \mu\tau,\ -3 + 2\mu\tau\}$, $R_{11} = \{-1 + 2\mu\tau,\ 3 - 4\mu\tau\}$, $R_{13} = \{1 + 2\mu\tau,\ -1 - 3\mu\tau,\ -5 + 3\mu\tau\}$, $R_{15} = \{1 - 3\mu\tau,\ 3 + 2\mu\tau,\ -3 + 3\mu\tau\}$.

   Set $c_5 = -1 + \mu\tau = -\mu\bar{\tau}$, $c_7 = 1 + \mu\tau = -(\mu\bar{\tau})^2$. $c_3 = -3 + \mu\tau = (\mu\bar{\tau})^3$, $c_{15} = 1 - 3\mu\tau = -(\mu\bar{\tau})^4$. Computing $Q_3 = c_3P, Q_{15} = c_{15}P$ requires 6**M**+4**S**. If $c_3$, $c_{15}$ are valued as others, the total cost will be more. $c_9$, $c_{11}$, $c_{13}$ are still not determined. Let $c_i \succ c_j$ denote $c_j = \mu\bar{\tau}c_i$ or $c_j = -\mu\bar{\tau}c_i$, i.e., $Q_i \succ Q_j$ denotes $Q_j = \mu\bar{\tau}Q_i$ or $Q_j = -\mu\bar{\tau}Q_i$. Recall that $R_9 = \{1 - 4\mu\tau,\ 3 + \mu\tau,\ -3 + 2\mu\tau\}$, $R_{11} = \{-1 + 2\mu\tau,\ 3 - 4\mu\tau\}$, $R_{13} = \{1 + 2\mu\tau,\ -1 - 3\mu\tau,\ -5 + 3\mu\tau\}$. List all $\succ$ relations for $c_9$, $c_{11}$, $c_{13}$ as follows.

   (a) $c_{11} = -1 + 2\mu\tau \succ c_9 = 3 + \mu\tau \succ c_{13} = -5 + 3\mu\tau$,

   (b) $c_9 = -3 + 2\mu\tau \succ c_{13} = -1 - 3\mu\tau$.

   Choose $c_9 = 3 + \mu\tau$, $c_{11} = -1 + 2\mu\tau = \mu\bar{\tau}(c_{11})$, and $c_{13} = -(\mu\bar{\tau})^2c_{11}$. It is the best choice to compute $Q_9$, $Q_{11}$, $Q_{13}$. Then $C = \{c_5 = -1 + \mu\tau, c_7 = 1 + \mu\tau, c_3 = -3 + \mu\tau, c_{15} = 1 - 3\mu\tau, c_{11} = -1 + 2\mu\tau, c_9 = 3 + \mu\tau, c_{13} = -5 + 3\mu\tau\}$. Thus our pre-computation scheme for the case of $w = 5$ as shown in Table 6 is determined.

3. For the case of $w = 6$, $R_1 = \{1,\ -1 - 5\mu\tau\}$, $R_3 = \{1 - 5\mu\tau,\ 3\}$, $R_5 = \{3 - 5\mu\tau,\ 5,\ -7 + 2\mu\tau\}$, $R_7 = \{-5 + 2\mu\tau,\ 5 - 5\mu\tau,\ 7\}$, $R_9 = \{-3 + 2\mu\tau,\ -5 - 3\mu\tau\}$, $R_{11} = \{-1 + 2\mu\tau,\ -3 - 3\mu\tau\}$, $R_{13} = \{1 + 2\mu\tau,\ -1 - 3\mu\tau\}$, $R_{15} = \{1 - 3\mu\tau,\ 3 + 2\mu\tau\}$, $R_{17} = \{3 - 3\mu\tau,\ 5 + 2\mu\tau,\ -7 + 4\mu\tau\}$, $R_{19} = \{5 - 3\mu\tau,\ -5 + 4\mu\tau,\ -7 - \mu\tau\}$, $R_{21} = \{-3 + 4\mu\tau,\ -5 - \mu\tau,\ 7 - 3\mu\tau\}$, $R_{23} = \{-1 + 4\mu\tau,\ -3 - \mu\tau\}$, $R_{25} = \{-1 - \mu\tau,\ 1 + 4\mu\tau\}$, $R_{27} = \{1 - \mu\tau,\ 3 + 4\mu\tau\}$, $R_{29} = \{3 - \mu\tau\}$, $R_{31} = \{3 - 6\mu\tau,\ 5 - \mu\tau,\ -7 + \mu\tau\}$.

   Notice that

$$\begin{aligned}
\mu\bar{\tau} &= 1 - \mu\tau,\\
(\mu\bar{\tau})^2 &= -1 - \mu\tau,\\
(\mu\bar{\tau})^3 &= -3 + \mu\tau,\\
(\mu\bar{\tau})^4 &= -1 + 3\mu\tau,\\
(\mu\bar{\tau})^5 &= 5 + \mu\tau.
\end{aligned}$$

First set $c_{27} = 1 - \mu\tau = \mu\bar{\tau}$, $c_{25} = -1 - \mu\tau = (\mu\bar{\tau})^2$, $c_{29} = 3 - \mu\tau = -(\mu\bar{\tau})^3$, $c_{15} = 1 - 3\mu\tau = -(\mu\bar{\tau})^4$, and $c_{21} = -5 - \mu\tau = -(\mu\bar{\tau})^5$. It is the best choice for calculating $Q_{15}$, $Q_{25}$, $Q_{27}$, $Q_{29}$, and $Q_{21}$ requiring 13**M**+9**S**.
$c_3$, $c_5$, $c_7$, $c_9$, $c_{11}$, $c_{13}$, $c_{17}$, $c_{19}$, $c_{23}$, $c_{31}$ are still not to be determined. List all $\succ$ relations for $c_3$, $c_5$, $c_7$, $c_9$, $c_{11}$, $c_{13}$, $c_{17}$, $c_{19}$, $c_{23}$, $c_{31}$ as follows.
(a) $c_3 = 3 \succ c_{17} = 3 - 3\mu\tau \succ c_{11} = -3 - 3\mu\tau$,
(b) $c_9 = -3 + 2\mu\tau \succ c_{13} = -1 - 3\mu\tau \succ c_{31} = -7 + \mu\tau$,
(c) $c_{11} = -1 + 2\mu\tau \succ c_{23} = -3 - \mu\tau \succ c_{19} = 5 - 3\mu\tau$,
(d) $c_{13} = 1 + 2\mu\tau \succ c_{31} = 5 - \mu\tau \succ c_5 = 3 - 5\mu\tau$,
(e) $c_{23} = -1 + 4\mu\tau \succ c_{19} = -7 - \mu\tau$,
(f) $c_5 = 5 \succ c_7 = 5 - 5\mu\tau$,
(g) $c_{31} = 5 - \mu\tau \succ c_5 = 3 - 5\mu\tau$,
(h) $c_7 = -5 + 2\mu\tau \succ c_3 = 1 - 5\mu\tau$.
There are four cases using 2 re-$\mu\bar{\tau}$ operations for computing $Q_3$, $Q_5$, $Q_7$, $Q_9$, $Q_{11}$, $Q_{13}$, $Q_{17}$, $Q_{19}$, $Q_{23}$, $Q_{31}$.

(a) $c_3 = 3 \succ c_{17} = 3 - 3\mu\tau \succ c_{11} = -3 - 3\mu\tau$,
$c_9 = -3 + 2\mu\tau \succ c_{13} = -1 - 3\mu\tau \succ c_{31} = -7 + \mu\tau$. Computing $Q_3$, $Q_9$, $Q_{11}$, $Q_{13}$, $Q_{17}$, $Q_{31}$ requires 2 $\mu\bar{\tau}$ operations, 2 re-$\mu\bar{\tau}$ operations, 1 $(P \pm Q)$-operation, and 1 $\tau$-affine operation which costs 28**M**+17**S**.
$c_5$, $c_7$, $c_{19}$, $c_{23}$ are still not determined. Notice that $R_5 = \{3-5\mu\tau,\ 5,\ -7+2\mu\tau\}$,
$R_7 = \{-5 + 2\mu\tau,\ 5 - 5\mu\tau,\ 7\}$,
$R_{19} = \{5 - 3\mu\tau,\ -5 + 4\mu\tau,\ -7 - \mu\tau\}$,
$R_{23} = \{-1 + 4\mu\tau,\ -3 - \mu\tau\}$.
The best choice is to choose $c_{23} = -1 + 4\mu\tau$, $c_{19} = -7 - \mu\tau = -\mu\bar{\tau}c_{23}$, $c_5 = -7 + 2\mu\tau$, $c_7 = 7$, where $Q_5$, $Q_7$, $Q_{19}$, and $Q_{23}$ can be computed as $Q_{23} = \mu\tau P - Q_{15}$, $Q_{19} = -\mu\bar{\tau}Q_{23}$, $Q_5 = \mu\tau P + Q_{31}$, $Q_7 = \mu\tau P - Q_{31}$. It requires 1 $(P \pm Q)$-operation, 1 $\mu\bar{\tau}$-operation, and 1 mixed addition which costs 25**M**+10**S**.
This pre-computation scheme totally requires 66**M**+36**S**.

(b) $c_3 = 3 \succ c_{17} = 3 - 3\mu\tau \succ c_{11} = -3 - 3\mu\tau$,
$c_{13} = 1 + 2\mu\tau \succ c_{31} = 5 - \mu\tau \succ c_5 = 3 - 5\mu\tau$.
It requires 32**M**+16**S** to compute $Q_3$, $Q_5$, $Q_{11}$, $Q_{13}$, $Q_{17}$, $Q_{31}$. $Q_7$, $Q_9$, $Q_{19}$, $Q_{23}$ are not computed.
$c_{23} = -3 - \mu\tau$, $c_{19} = 5 - 3\mu\tau = -\mu\bar{\tau}c_{23}$, $c_7 = -5 + 2\mu\tau$, $c_9 = -3 + 2\mu\tau$ ($c_3P, c_9P$ are computed together) is one of the best choices which requires 25**M**+10**S**. It costs 70**M**+35**S** totally.

(c) $c_{11} = -1 + 2\mu\tau \succ c_{23} = -3 - \mu\tau \succ c_{19} = 5 - 3\mu\tau$,
$c_{13} = 1 + 2\mu\tau \succ c_{31} = 5 - \mu\tau \succ c_5 = 3 - 5\mu\tau$.
It requires 32**M**+16**S** to compute $Q_5$, $Q_{11}$, $Q_{13}$, $Q_{19}$, $Q_{23}$, $Q_{31}$.
$c_3 = 3$, $c_{17} = 3 - 3\mu\tau = \mu\bar{\tau}c_3$, $c_9 = -3 + 2\mu\tau$, $c_7 = -5 + 2\mu\tau$ is the best choice. It requires 1 $(P \pm Q)$-operation ($c_3P, c_9P$ are computed together), 1 $\mu\bar{\tau}$-operation, and 1 mixed addition which costs 25**M**+10**S**. It totally requires 70**M**+35**S**.

(d) $c_9 = -3 + 2\mu\tau \succ c_{13} = -1 - 3\mu\tau \succ c_{31} = -7 + \mu\tau$.
$c_{11} = -1 + 2\mu\tau \succ c_{23} = -3 - \mu\tau \succ c_{19} = 5 - 3\mu\tau$,

It requires 32**M**+16**S** to compute $Q_9$, $Q_{11}$, $Q_{13}$, $Q_{19}$, $Q_{23}$, $Q_{31}$. $Q_3$, $Q_5$, $Q_7$, $Q_{17}$ are not computed.
$c_3 = 3$, $c_{17} = 3 - 3\mu\tau = \mu\bar{\tau}c_3$, $c_5 = -7 + 2\mu\tau$, $c_7 = 7$ is the best choice which requires 25**M**+10**S**.
It totally costs 70**M**+35**S**.

If we use only one re-$\mu\bar{\tau}$ operation or do not use re-$\mu\bar{\tau}$ operation for computing $Q_3$, $Q_5$, $Q_7$, $Q_9$, $Q_{11}$, $Q_{13}$, $Q_{17}$, $Q_{19}$, $Q_{21}$, $Q_{23}$, $Q_{31}$, total costs will not be less than the Case *a*).
Choosing $c_9 = -3+2\mu\tau$, $c_3 = 3$, $c_{13} = -1-3\mu\tau$, $c_{31} = -7+\mu\tau$, $c_{17} = 3-3\mu\tau$, $c_{11} = -3 - 3\mu\tau$, $c_{23} = -1 + 4\mu\tau$, $c_{19} = -7 - \mu\tau$, $c_5 = 5$, $c_7 = 5 - 5\mu\tau$, we have $C = \{c_{27} = 1 - \mu\tau, c_{25} = -1 - \mu\tau, c_{29} = 3 - \mu\tau, c_{15} = 1 - 3\mu\tau, c_{21} = -5 - \mu\tau, c_{11} = -1 + 2\mu\tau, c_{23} = -1 + 4\mu\tau, c_{19} = -7 - \mu\tau, c_9 = -3 + 2\mu\tau, c_3 = 3, c_{13} = -1 - 3\mu\tau, c_{31} = -7 + \mu\tau, c_{17} = 5 + 2\mu\tau, c_5 = 5, c_7 = 5 - 5\mu\tau\}$. The pre-computation scheme for the case of $w = 6$ shown as Table 6 is our best choice.

## Acknowledgments

This work is supported in part by the National Nature Science Foundation of China under No. 61502487 and No. 61772515, and the National 973 Project of China under No. 2013CB834205.

## References

1. Koblitz N.: CM-curves with good cryptographic properties, in Proc. 11th Annu. Int. Cryptol. Conf. Adv. Cryptol., pp. 279-287, 1992.
2. National Institute of Standards and Technology(NIST). Digital signature standard(DSS). FIPS PUB 186-4. 2013.
3. Solinas J.: Efficient arithmetic on Koblitz curves, Des., Codes Cryptography, vol. 19, pp. 195-249, 2000.
4. Blake I., Murty V., and Xu G.: A note on window $\tau$-NAF algorithm, Inf. Process. Lett., vol. 95, no. 5, pp. 496-502, 2005.
5. Blake I., Murty V., and Xu G.: Nonadjacent radix-$\tau$ expansions of integers in Euclidean imaginary quadratic number fields, Canadian J. Math., vol. 60, pp. 1267-1282, 2008.
6. Trost W.R. and Xu G.: On the optimal pre-computation of window $\tau$NAF for Koblitz curves. IEEE transactions on computers. VOl. 65, No. 9. pp. 2918-2924, september 2016.
7. Hankerson D., Menezes A., and Vanstone S.: Guide to elliptic curve cryptography. New York, NY, USA: Springer-Verlag, 2004.
8. Oliveira T., López J., Aranha D.F., and Rodríguez-Henríquez F.: Two is the fastest prime: Lambda coordinates for binary elliptic curves, J. Cryptography Eng. vol. 4, no. 1, pp. 3-7, 2014.
9. Avanzi R.M., Dimitrov V.S., Doche C., and Sica F.: Extending scalar multiplication using double bases. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 130-144. Springer, Heidelberg, 2006.

10. Doche C., Kohel D.R., and Sica F.: Double Base Number System for multi scalar multiplications. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 502-517. Springer, Heidelberg, 2009.
11. Koblitz N.: $p$-adic numbers, $p$-adic analysis, and zeta-functions. New York, NY, USA: Springer, 1996.
12. López J., Dahab R.: Improved algorithms for elliptic curve arithmetic in $GF(2^n)$, in Proc. Selected Areas Cryptography, pp. 201-212, 1998.
13. Longa P., Gebotys C.: Novel precomputation schemes for elliptic curve cryptosystems, ACNS 2009, LNCS, vol. 5536, pp. 71-88, 2009.
14. Bernstein D. J., and Lange T.: Explicit-formulas database, `http://hyperelliptic.org/EFD/`
15. Scott, M.:: MIRACL-Multiprecision integer and rational arithmetic cryptographic library, C/C++ Library, `ftp://ftp.computing.dcu.ie/pub/crypto/miracl.zip`