Pre-Computation Scheme of Window τNAF for Koblitz Curves Revisited

Wei Yu[1(✉)] and Guangwu Xu[2,3(✉)]

[1] State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China
yuwei_1_yw@163.com
[2] Key Laboratory of Cryptologic Technology and Information Security of Ministry of Education, Qingdao, Shandong, 266237, China
gxu4sdq@sdu.edu.cn
[3] School of Cyber Science and Technology, Shandong University, Qingdao, Shandong, 266237, China.

Abstract. Let $E_a/\mathbb{F}_2 : y^2 + xy = x^3 + ax^2 + 1$ be a Koblitz curve. The window τ-adic non-adjacent form (window τNAF) is currently the standard representation system to perform scalar multiplications on $E_a/\mathbb{F}_{2^m}$ utilizing the Frobenius map τ. This work focuses on the pre-computation part of scalar multiplication. We first introduce $\mu\bar{\tau}$-operations where $\mu = (-1)^{1-a}$ and $\bar{\tau}$ is the complex conjugate of τ. Efficient formulas of $\mu\bar{\tau}$-operations are then derived and used in a novel pre-computation scheme. Our pre-computation scheme requires 6**M**+6**S**, 18**M**+17**S**, 44**M**+32**S**, and 88**M**+62**S** ($a = 0$) and 6**M**+6**S**, 19**M**+17**S**, 46**M**+32**S**, and 90**M**+62**S** ($a = 1$) for window τNAF with widths from 4 to 7 respectively. It is about two times faster, compared to the state-of-the-art technique of pre-computation in the literature. The impact of our new efficient pre-computation is also reflected by the significant improvement of scalar multiplication. Traditionally, window τNAF with width at most 6 is used to achieve the best scalar multiplication. Because of the dramatic cost reduction of the proposed pre-computation, we are able to increase the width for window τNAF to 7 for a better scalar multiplication. This indicates that the pre-computation part becomes more important in performing scalar multiplication. With our efficient pre-computation and the new window width, our scalar multiplication runs in at least 85.2% the time of Kohel's work (Eurocrypt'2017) combining the best previous pre-computation. Our results push the scalar multiplication of Koblitz curves, a very well-studied and long-standing research area, to a significant new stage.

Keywords: Elliptic curve cryptography, Koblitz curve, Scalar multiplication, Window τNAF, Pre-computation.

1 Introduction

Elliptic curve cryptography has drawn extensive attention from the literature [25, 30]. The family of Koblitz curves, proposed by Koblitz in [12], are non-supersingular curves defined over $\mathbb{F}_2$. The arithmetic of Koblitz curves has been of theoretical and practical significance since the start of elliptic curve cryptography. 4 Koblitz curves were recommended to be used in digital signature, key-establishment, and key management by National Institute of Standards and Technology (NIST) FIPS 186-5(draft) [21]–"digital signature standard" (October of 2019), NIST special publication 800-56A (revision 3)–"recommendation for pair-wise key-establishment schemes using discrete logarithm cryptography" [3] (April of 2018), and NIST special publication 800-57 Part 1 (revision 5)–"recommendation for key management, part 1: general" [2] (May of 2020) respectively. These indicate that Koblitz curves can still be useful in practice.

Koblitz 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 $160 < m < 2000$, $\frac{|E_0(\mathbb{F}_{2^m})|}{|E_0(\mathbb{F}_2)|}$ is a prime when $m = 233, 239, 277, 283, 349, 409, 571, 1249$, and 1913, and $\frac{|E_1(\mathbb{F}_{2^m})|}{|E_1(\mathbb{F}_2)|}$ is a prime when $m = 163, 283, 311, 331, 347, 359, 701, 1153, 1597$, and 1621. Four Koblitz curves with $a = 0$ have been recommended by NIST [2, 3, 21]: K-233($a = 0$), K-283($a = 0$), K-409($a = 0$), and K-571($a = 0$). Koblitz curves with $a = 1$ over $\mathbb{F}_{2^{163}}$, $\mathbb{F}_{2^{283}}$, $\mathbb{F}_{2^{359}}$, and $\mathbb{F}_{2^{701}}$ denoted by K1-163(for legacy-use only), K1-283, K1-359, and K1-701 respectively are also investigated in this work.

The Frobenius map τ 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 τ can be interpreted as a complex number 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 for all $P \in M$.

Koblitz [12] 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 [27], Solinas further developed an extremely efficient window τNAF to compute nP. Refinements and extensions of Solinas' method were obtained by Blake, Murty and Xu [5, 6].

The procedure of window τNAF can be described as four steps [5, 28].

1. Reduction. Find a suitable $\rho \in \mathbb{Z}[\tau]$ satisfying $\rho \equiv n \pmod{\delta}$.
2. Window τNAF 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 τNAF of n is the following sparse τ expansion of its reduction ρ:

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

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 τNAF and pre-computation.

Pre-computation plays a significant role in improving the efficiency of scalar multiplications using window τNAF. For window τ-NAF with widths w, $2^{w-2}-1$ pre-computed points require to be stored in memory. Several ways of designing pre-computations have been proposed by Solinas [27], Blake, Murty and Xu [5], and Hankerson, Menezes, and Vanstone [10]. In fact, [5] established a framework under which pre-computations for window τNAF can be made more flexible. This framework also enables a rigorous proof of termination of window τNAF. In [6], the authors investigated fast scalar multiplications for larger family of elliptic curves by developing non-adjacent radix-τ expansions for integers in other Euclidean imaginary quadratic number fields. Later, Trost and Xu [28] introduced an optimal pre-computation of window τNAF that improves previous results. However, the main objective of the pre-computation in [28] is its mathematically natural and clean forms. The optimality is based on the fact that it requires $2^{w-2} - 1$ point additions and two evaluations of the Frobenius map τ. They employed λ-coordinates [24] to achieve an improvement on performance of scalar multiplication and provided a convenient structure for further work.

In 2017, Kohel introduced a twisted μ_4-normal form elliptic curve over a binary field for its efficiency in [15]. Kohel proved that twisted μ_4-normal form elliptic curves cover all the elliptic curves over binary fields recommended by NIST. A Koblitz curve using twisted μ_4-normal form is called a μ_4-Koblitz curve. Because of its promising computational advantage, it is of great interest to consider the use of μ_4-Koblitz curves in the window τNAF, especially for the pre-computation part.

Let us summarize the cost of existing pre-computation schemes for window τ-NAF with widths $w = 4$, 5, and 6 on μ_4-Koblitz curves (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 [27] covers $w = 4$ and 5 only. The corresponding costs are 15**M**+15**S** and

38**M**+38**S** with $a = 0$ and those are 18**M**+15**S** and 45**M**+38**S** with $a = 1$. In [10], $w = 4$, 5, and 6 are considered. The corresponding costs are 15**M**+15**S**, 40**M**+35**S**, and 89**M**+67**S** with $a = 0$ and those are 18**M**+15**S**, 47**M**+35**S**, and 104**M**+67**S** with $a = 1$. The pre-computation scheme constructed in [28] has improved the above costs to 15**M**+12**S**, 39**M**+20**S**, and 87**M**+36**S** with $a = 0$ and 18**M**+12**S**, 46**M**+20**S**, and 102**M**+36**S** with $a = 1$ for $w = 4$, 5, and 6.

Our contributions The main purpose of this work is twofold. Firstly, we develop an efficient way of calculating pre-computation for the window τNAF on Koblitz curves; and secondly, we propose to use a bigger width in the window τNAF together with our pre-computation to achieve a significant speedup on scalar multiplication. By using a μ_4-Koblitz curve, 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 τ and P be a rational point on a Koblitz curve. Both Avanzi, Dimitrov, Doche, and Sica [1] and Doche, Kohel, and Sica [8] used complex multiplication $\bar{\tau}P$ in double-base representation to speed up scalar multiplication. Inspired by their elegant results, we introduce a new radix $\mu\bar{\tau}$. Under this radix, we design new formulas for $\mu\bar{\tau}P$ which only requires 2**M**+2**S**. Trost and Xu proved that one point addition is necessary for computing each pre-computation point $Q_i,\ i \in \{3, 5, \ldots, 2^{w-1} - 1\}$ [28]. We use $\mu\bar{\tau}$-operations to replace point additions or mixed additions in pre-computation scheme. As the cost of one full addition is 7**M**+2**S** and that of one mixed addition is 6**M**+2**S** for $a = 0$ and those are 8**M**+2**S** and 7**M**+2**S** respectively for $a = 1$, our formulas of $\mu\bar{\tau}P$ save quite a few field operations. Our formulas for $\mu\bar{\tau}P$ are part of doubling formulas, which may lead to a simplicity of the implementation.
2. We propose a plane search to generate R_i whose elements are with the form of $g + h\mu\tau$. To take full advantage of our $\mu\bar{\tau}$-operations, we choose one suitable $c_i \in R_i$ for each $i \in I_w$ generated by the plane search. A novel pre-computation scheme is developed to save more field operations. Our pre-computation scheme requires 6**M**+6**S**, 18**M**+17**S**, 44**M**+32**S**, and 88**M**+62**S** ($a = 0$) and 6**M**+6**S**, 19**M**+17**S**, 46**M**+32**S**, and 90**M**+62**S** ($a = 1$) for window τNAF with widths from 4 to 7 respectively. The cost of Solinas' pre-computation scheme, that of Hankerson, Menezes, and Vanstone's pre-computation scheme, that of Trost and Xu's pre-computation scheme, and that our pre-computation scheme on μ_4-Koblitz curves with $a = 0$ and $a = 1$ are shown in Table 1. The practical implementations show that our pre-computation is two times faster than Trost and Xu's pre-computation and are consistent with our theoretical analysis.
3. In window τNAF, a bigger window width corresponds to a sparser τ expansion for scalar multiplication. However, one should not make the width too big as it would increase the pre-computation cost and affect the overall performance. Currently, the state-of-the-art pre-computation scheme suggests to use width at most 6 to achieve the best efficiency of scalar multiplication. Our pre-computation reduces the cost by half in most

practical cases, namely, scheme with width 7 is about the same as the cost of existing pre-computation scheme with width 6. This allows us to use a bigger window width (e.g., 7) to get a faster scalar multiplication. The balance between the pre-computation part and the other part of scalar multiplication shows that the pre-computation takes a bigger ratio of scalar multiplication than before. This is useful especially for scalar multiplication with unfixed point. Constant-time scalar multiplication using our novel pre-computation on a μ_4-Koblitz curve saves up to 33.5% compared to that using Trost and Xu's pre-computation in López-Dahab (LD) coordinates [20], saves up to 28.6% compared to Trost and Xu's original work [28], and saves up to 14.8% compared to Kohel's work [15] combining Trost and Xu's pre-computation. It is about 4 times faster compared to the state-of-the-art non-pre-computation-based constant-time scalar multiplication in LD coordinates, about 4 times faster in λ-coordinates, and over 3 times faster on a μ_4-Koblitz curve.

Table 1. Cost of pre-computations on a μ_4-Koblitz curve

		$w=4$	$w=5$	$w=6$
$a=0$	Solinas [27]	15**M**+15**S**	38**M**+38**S**	-
	Hankerson,Menezes,Vanstone [10]	15**M**+15**S**	40**M**+35**S**	89**M**+67**S**
	Trost, Xu [28]	15**M**+12**S**	39**M**+20**S**	87**M**+36**S**
	Ours	6**M**+6**S**	18**M**+17**S**	44**M**+32**S**
$a=1$	Solinas [27]	18**M**+15**S**	45**M**+38**S**	-
	Hankerson,Menezes,Vanstone [10]	18**M**+15**S**	47**M**+35**S**	104**M**+67**S**
	Trost, Xu [28]	18**M**+12**S**	46**M**+20**S**	102**M**+36**S**
	Ours	6**M**+6**S**	19**M**+17**S**	47**M**+32**S**

This paper is organized as follows. In Section 2, we present previous pre-computation schemes of window τNAF 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 and compare scalar multiplications in experimental implementations. Finally, we discuss our pre-computation in Section 7.

2 Preliminary

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

2.1 Determine $\tau^w|(g + h\tau)$

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 Lucas sequence in [27] or by the approach suggested in [6] based on Hensel's lifting procedure [13].

Using Lucas sequence or Hensel's lifting algorithm, we get $s_2 = 2\mu$, $s_3 = 6\mu$, $s_4 = 6\mu$, $s_5 = 6\mu$, $s_6 = 38\mu$, $s_7 = 38\mu$, $s_8 = 166\mu$, $s_9 = 422\mu$, and $s_{10} = 934\mu$. When $w \geq 2$, $s_w \equiv 0 \pmod 2$ and $s_w/2$ is odd.

It has been proved in [6, 27] that for each positive integer w,

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

2.2 Costs of Point Operations on Koblitz Curves

We summarize the costs of point operations on Koblitz curves using LD coordinates [20], λ-coordinates [24], and those on a μ_4-Koblitz curve [15] shown as Table 2. We neglect the cost of a field addition since it involves only bitwise XORs.

Table 2. Costs of point operations on Koblitz curves

Coordinates	$\tau(P)$	τ-affine operation	addition	mixed addition*
LD coordinates [17, 20]	3**S**	2**S**	13**M**+4**S**	8**M**+5**S**
λ-coordinates [24]	3**S**	2**S**	11**M**+2**S**	8**M**+2**S**
μ_4-Koblitz curve ($a = 0$) [14]	4**S**	3**S**	7**M**+2**S**	6**M**+2**S**
μ_4-Koblitz curve ($a = 1$) [15, 18]	4**S**	3**S**	8**M**+2**S**	7**M**+2**S**

* Let P, Q be rational points in the main subgroup M. $\tau(P)$ is denoted by τ-affine operation or $P + Q$ is denoted by mixed addition when the Z-coordinate of P is 1 using LD coordinates, that is 1 using λ-coordinates, or X_2-coordinate of P is 1 on a μ_4-Koblitz curve.

Let $a \in \{0, 1\}$. A Koblitz curve $y^2 + xy = x^3 + ax^2 + 1$ can be translated into a μ_4-Koblitz curve $X_0^2 + X_2^2 = X_1X_3 + aX_0X_2$, $X_1^2 + X_3^2 = X_0X_2$ via the map $(x, y) \mapsto (x^2 : x^2 + y : 1 : x^2 + y + x)$ and the inverse is $(X_0 : X_1 : X_2 : X_3) \mapsto (X_1 + X_3 : X_0 + X_1 : X_2)$ [15]. The identity of a μ_4-Koblitz curve is $(1 : 1 : 0 : 1)$. The inverse morphism is $[-1](X_0 : X_1 : X_2 : X_3) = (X_0 : X_3 : X_2 : X_1)$. The projective point $(X_0 : X_1 : X_2 : X_3)$ on a μ_4-Koblitz curve can be translated into an affine point $(\frac{X_0}{X_2} : \frac{X_1}{X_2} : 1 : \frac{X_3}{X_2})$. $\tau(X_0 : X_1 : X_2 : X_3) = (X_0^2 : X_1^2 : X_2^2 : X_3^2)$ and $\tau^2(P) + 2P = \mu\tau(P)$ where $\mu = (-1)^{1-a}$. On a μ_4-Koblitz curve, a τ-operation requires 4**S** and a τ-affine operation requires 3**S**.

In particular, μ_4-Koblitz curve with $a = 0$ corresponds to the curve given in Theorem 4 of [14] with $c = 1$. In the case of $a = 0$, one full point addition requires 7**M**+2**S**, one mixed addition requires 6**M**+2**S**, and one point addition with both affine points (X_2-components of both summands can be set to 1) requires 5**M**+2**S** [14]. In the case of $a = 1$, one full point addition requires 8**M**+2**S**, one mixed addition requires 7**M**+2**S**, and one point addition with both affine points requires 6**M**+2**S** [15, 18].

The LD coordinates system and λ-coordinates system, proposed by López and Dahab [20] and by Oliveira, López, Aranha, and Rodríguez-Henríquez [24] respectively, are also efficient coordinate systems for binary elliptic curves. In Appendixes B and C, we will utilize our pre-computation scheme on Koblitz curves using LD coordinates and λ-coordinates.

2.3 Previous Pre-Computation Schemes

We will consider the efficiency of pre-computation schemes on a μ_4-Koblitz curve.

Solinas' pre-computation [27] Solinas suggested an efficient design of the pre-computation and gave an example shown in Table 3. Computing $Q_3 = -P + \tau^2 P$ requires one point addition with both affine points and two τ-affine operations at the total cost of (5**M**+2**S**)+6**S**. The other costs are similarly computed in Table 3 and in the following pre-computation schemes. The costs of Solinas' pre-computation are 15**M**+15**S** and 38**M**+38**S** with $a = 0$ and 18**M**+15**S** and 45**M**+38**S** with $a = 1$ for window τNAF with widths 4 and 5 respectively.

Table 3. Pre-computation scheme in [27]

	$a = 0$	$a = 1$	cost($a = 0$)
$w = 4$			15**M**+15**S**
	$Q_3 = -P + \tau^2 P(c_3 = -\tau - 3)$	$Q_3 = -P + \tau^2 P(c_3 = \tau - 3)$	(5**M**+2**S**)+6**S**
	$Q_5 = P + \tau^2 P(c_5 = -\tau - 1)$	$Q_5 = P + \tau^2 P(c_5 = \tau - 1)$	5**M**+2**S**
	$Q_7 = -P + \tau^3 P(c_7 = -\tau + 1)$	$Q_7 = -P - \tau^3 P(c_7 = \tau + 1)$	(5**M**+2**S**)+3**S**
$w = 5$			38**M**+38**S**
	$Q_3 = -P + \tau^2 P(c_3 = -\tau - 3)$	$Q_3 = -P + \tau^2 P(c_3 = \tau - 3)$	(5**M**+2**S**)+6**S**
	$Q_5 = P + \tau^2 P(c_5 = -\tau - 1)$	$Q_5 = P + \tau^2 P(c_5 = \tau - 1)$	5**M**+2**S**
	$Q_7 = -P + \tau^3 P(c_7 = -\tau + 1)$	$Q_7 = -P - \tau^3 P(c_7 = \tau + 1)$	(5**M**+2**S**)+3**S**
	$Q_9 = P + \tau^3 Q_5(c_9 = -2\tau - 3)$	$Q_9 = P - \tau^3 Q_5(c_9 = 2\tau - 3)$	(6**M**+2**S**)+12**S**
	$Q_{11} = -\tau^2 Q_5 - P(c_{11} = -2\tau - 1)$	$Q_{11} = -\tau^2 Q_5 - P(c_{11} = 2\tau - 1)$	6**M**+2**S**
	$Q_{13} = -\tau^2 Q_5 + P(c_{13} = -2\tau + 1)$	$Q_{13} = -\tau^2 Q_5 + P(c_{13} = 2\tau + 1)$	6**M**+2**S**
	$Q_{15} = -P + \tau^4 P(c_{15} = 3\tau + 1)$	$Q_{15} = -P + \tau^4 P(c_{15} = -3\tau + 1)$	(5**M**+2**S**)+3**S**

Hankerson, Menezes, and Vanstone's pre-computation [10] Hankerson, Menezes, and Vanstone presented an improved design of pre-computation shown in Table 4. The costs of Hankerson, Menezes, and Vanstone's pre-computation are 15**M**+15**S**, 40**M**+35**S**, and 89**M**+75**S** with $a = 0$ and 18**M**+15**S**, 47**M**+35**S**, and 104**M**+75**S** with $a = 1$ for window τNAF with widths 4, 5, and 6 respectively.

Trost and Xu's pre-computation [28] Trost and Xu proposed a mathematically natural and clean form of pre-computation. The pre-computation requires the least number of point additions and τ evaluations. We include their pre-computation scheme for window τNAF with widths 4, 5, and 6 in Table 5. The costs are 15**M**+12**S**, 39**M**+20**S**, and 87**M**+36**S** with $a = 0$ and 18**M**+12**S**, 46**M**+20**S**, and 102**M**+36**S** with $a = 1$.

Trost and Xu did not get into field arithmetic details to speed up the pre-computation. Our main objective of this paper is to design a novel pre-computation and efficient formulas to achieve a great saving of scalar multiplication. To implement scalar multiplication, Montgomery trick may be useful.

Table 4. Pre-computation scheme in [10]

	$a=0$	$a=1$	cost$(a=0)$
$w=4$			**15M+15S**
	$Q_3=-P+\tau^2P(c_3=-\tau-3)$	$Q_3=-P+\tau^2P(c_3=\tau-3)$	(**5M+2S**)+6**S**
	$Q_5=P+\tau^2P(c_5=-\tau-1)$	$Q_5=P+\tau^2P(c_5=\tau-1)$	**5M+2S**
	$Q_7=-P+\tau^3P(c_7=-\tau+1)$	$Q_7=-P-\tau^3P(c_7=\tau+1)$	(**5M+2S**)+3**S**
$w=5$			**40M+35S**
	$Q_3=-P+\tau^2P(c_3=-\tau-3)$	$Q_3=-P+\tau^2P(c_3=\tau-3)$	(**5M+2S**)+6**S**
	$Q_5=P+\tau^2P(c_5=-\tau-1)$	$Q_5=P+\tau^2P(c_5=\tau-1)$	**5M+2S**
	$Q_7=-P+\tau^3P(c_7=-\tau+1)$	$Q_7=-P-\tau^3P(c_7=\tau+1)$	(**5M+2S**)+3**S**
	$Q_9=P+\tau^3Q_5(c_9=-2\tau-3)$	$Q_9=P-\tau^3Q_5(c_9=2\tau-3)$	(**6M+2S**)+12**S**
	$Q_{11}=-\tau^2Q_5-P(c_{11}=-2\tau-1)$	$Q_{11}=-\tau^2Q_5-P(c_{11}=2\tau-1)$	**6M+2S**
	$Q_{13}=-\tau^2Q_5+P(c_{13}=-2\tau+1)$	$Q_{13}=-\tau^2Q_5+P(c_{13}=2\tau+1)$	**6M+2S**
	$Q_{15}=-Q_5+\tau^2Q_5(c_{15}=3\tau+1)$	$Q_{15}=-Q_5+\tau^2Q_5(c_{15}=-3\tau+1)$	**7M+2S**
$w=6$			**89M+75S**
	$Q_{23}=-P-\tau^3P(c_{23}=\tau-3)$	$Q_{23}=-P+\tau^3P(c_{23}=-\tau-3)$	(**5M+2S**)+9**S**
	$Q_{25}=P-\tau^3P(c_{25}=\tau-1)$	$Q_{25}=P+\tau^3P(c_{25}=-\tau-1)$	**5M+2S**
	$Q_{27}=-P-\tau^2P(c_{27}=\tau+1)$	$Q_{27}=-P-\tau^2P(c_{27}=-\tau+1)$	**5M+2S**
	$Q_{29}=P-\tau^2P(c_{29}=\tau+3)$	$Q_{29}=P-\tau^2P(c_{29}=-\tau+3)$	**5M+2S**
	$Q_3=\tau^2Q_{25}-P(c_3=3)$	$Q_3=\tau^2Q_{25}-P(c_3=3)$	(**6M+2S**)+8**S**
	$Q_5=\tau^2Q_{25}+P(c_5=5)$	$Q_5=\tau^2Q_{25}+P(c_5=5)$	**6M+2S**
	$Q_7=-\tau^3Q_{27}-P(c_7=-2\tau-5)$	$Q_7=\tau^3Q_{27}-P(c_7=2\tau-5)$	(**6M+2S**)+12**S**
	$Q_9=-\tau^3Q_{27}+P(c_9=-2\tau-3)$	$Q_9=\tau^3Q_{27}+P(c_9=2\tau-3)$	**6M+2S**
	$Q_{11}=\tau^2Q_{27}-P(c_{11}=-2\tau-1)$	$Q_{11}=\tau^2Q_{27}-P(c_{11}=2\tau-1)$	**6M+2S**
	$Q_{13}=\tau^2Q_{27}+P(c_{13}=-2\tau+1)$	$Q_{13}=\tau^2Q_{27}+P(c_{13}=2\tau+1)$	**6M+2S**
	$Q_{15}=-\tau^2Q_{27}+Q_{27}(c_{15}=3\tau+1)$	$Q_{15}=-\tau^2Q_{27}+Q_{27}(c_{15}=-3\tau+1)$	**7M+2S**
	$Q_{17}=-\tau^2Q_{27}+Q_{29}(c_{17}=3\tau+3)$	$Q_{17}=-\tau^2Q_{27}+Q_{29}(c_{17}=-3\tau+3)$	**7M+2S**
	$Q_{19}=-\tau^2Q_3-P(c_{19}=3\tau+5)$	$Q_{19}=-\tau^2Q_3-P(c_{19}=-3\tau+5)$	(**6M+2S**)+8**S**
	$Q_{21}=\tau^2Q_{29}+P(c_{21}=-4\tau-3)$	$Q_{21}=\tau^2Q_{29}+P(c_{21}=4\tau-3)$	(**6M+2S**)+8**S**
	$Q_{31}=\tau^2Q_{25}+Q_{27}(c_{31}=\tau+5)$	$Q_{31}=\tau^2Q_{25}+Q_{27}(c_{31}=-\tau+5)$	**7M+2S**

Table 5. Pre-computation scheme in [28]

	$a=0$	$a=1$	cost$(a=0)$
$w=4$			**15M+12S**
	$Q_5=-P-\tau P(c_5=-\tau-1)$	$Q_5=-P+\tau P(c_5=\tau-1)$	(**5M+2S**)+3**S**
	$Q_7=P-\tau P(c_7=-\tau+1)$	$Q_7=P+\tau P(c_7=\tau+1)$	**5M+2S**
	$Q_3=-P+\tau^2P(c_3=-\tau-3)$	$Q_3=-P+\tau^2P(c_3=\tau-3)$	(**5M+2S**)+3**S**
$w=5$			**39M+20S**
	$Q_5=-P-\tau P(c_5=-\tau-1)$	$Q_5=-P+\tau P(c_5=\tau-1)$	(**5M+2S**)+3**S**
	$Q_7=P-\tau P(c_7=-\tau+1)$	$Q_7=P+\tau P(c_7=\tau+1)$	**5M+2S**
	$Q_3=-P+\tau^2P(c_3=-\tau-3)$	$Q_3=-P+\tau^2P(c_3=\tau-3)$	(**5M+2S**)+3**S**
	$Q_9=Q_3-\tau P(c_9=-2\tau-3)$	$Q_9=Q_3+\tau P(c_9=2\tau-3)$	**6M+2S**
	$Q_{11}=Q_5-\tau P(c_{11}=-2\tau-1)$	$Q_{11}=Q_5+\tau P(c_{11}=2\tau-1)$	**6M+2S**
	$Q_{13}=Q_7-\tau P(c_{13}=-2\tau+1)$	$Q_{13}=Q_7+\tau P(c_{13}=2\tau+1)$	**6M+2S**
	$Q_{15}=-Q_{11}+\tau P(c_{15}=3\tau+1)$	$Q_{15}=-Q_{11}-\tau P(c_{15}=-3\tau+1)$	**6M+2S**
$w=6$			**87M+36S**
	$Q_{27}=P+\tau P(c_{27}=\tau+1)$	$Q_{27}=P-\tau P(c_{27}=-\tau+1)$	(**5M+2S**)+3**S**
	$Q_{25}=-P+\tau P(c_{25}=\tau-1)$	$Q_{25}=-P-\tau P(c_{25}=-\tau-1)$	**5M+2S**
	$Q_{29}=P-\tau^2P(c_{29}=\tau+3)$	$Q_{29}=P-\tau^2P(c_{29}=-\tau+3)$	(**5M+2S**)+3**S**
	$Q_3=Q_{29}-\tau P(c_3=3)$	$Q_3=Q_{29}+\tau P(c_3=3)$	**6M+2S**
	$Q_9=-Q_{29}-\tau P(c_9=-2\tau-3)$	$Q_9=-Q_{29}+\tau P(c_9=2\tau-3)$	**6M+2S**
	$Q_{31}=Q_3-\tau^2P(c_{31}=\tau+5)$	$Q_{31}=Q_3-\tau^2P(c_{31}=-\tau+5)$	**6M+2S**
	$Q_5=Q_{31}-\tau P(c_5=5)$	$Q_5=Q_{31}+\tau P(c_5=5)$	**6M+2S**
	$Q_7=-Q_{31}-\tau P(c_7=-2\tau-5)$	$Q_7=-Q_{31}+\tau P(c_7=2\tau-5)$	**6M+2S**
	$Q_{11}=-Q_{27}-\tau P(c_{11}=-2\tau-1)$	$Q_{11}=-Q_{27}+\tau P(c_{11}=2\tau-1)$	**6M+2S**
	$Q_{13}=-Q_{25}-\tau P(c_{13}=-2\tau+1)$	$Q_{13}=-Q_{25}+\tau P(c_{13}=2\tau+1)$	**6M+2S**
	$Q_{15}=-Q_{11}+\tau P(c_{15}=3\tau+1)$	$Q_{15}=-Q_{11}-\tau P(c_{15}=-3\tau+1)$	**6M+2S**
	$Q_{17}=-Q_9+\tau P(c_{17}=3\tau+3)$	$Q_{17}=-Q_9-\tau P(c_{17}=-3\tau+3)$	**6M+2S**
	$Q_{19}=-Q_7+\tau P(c_{19}=3\tau+5)$	$Q_{19}=-Q_7-\tau P(c_{19}=-3\tau+5)$	**6M+2S**
	$Q_{21}=-Q_{17}-\tau P(c_{21}=-4\tau-3)$	$Q_{21}=-Q_{17}+\tau P(c_{21}=4\tau-3)$	**6M+2S**
	$Q_{23}=-Q_3+\tau P(c_{23}=\tau-3)$	$Q_{23}=-Q_3-\tau P(c_{23}=-\tau-3)$	**6M+2S**

2.4 Montgomery Trick

Montgomery trick [7] computes simultaneously the inversions of n elements. It requires one inversion and $3(n-1)$ multiplications. Montgomery trick is powerful to translate points in projective coordinates to those in affine coordinates shown as Algorithm 1. For n points $(X_{0i} : X_{1i} : X_{2i} : X_{3i}), 1 \leq i \leq n$, we use Montgomery trick to compute X_{2i}^{-1}, and then compute $(\frac{X_{0i}}{X_{2i}} : \frac{X_{1i}}{X_{2i}} : 1 : \frac{X_{3i}}{X_{2i}})$. This trick translates n projective points on a μ_4-Koblitz curve to those in affine coordinates on a μ_4-Koblitz curve. When projective points are converted to affine points, we replace full point addition with mixed point addition to get a higher efficiency of scalar multiplication when the ratio of $\mathbf{I}/\mathbf{M}$ is not too high.

Algorithm 1 Montgomery trick [7]

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 from 2 to n
   $c_i \leftarrow c_{i-1} \cdot a_i$
3. $d \leftarrow c_n^{-1}$
4. for i from 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

In the next section, we will propose new formulas on a μ_4-Koblitz curve to design an efficient pre-computation scheme.

3 New Formulas on μ_4-Koblitz Curves

Let $P(X_0 : X_1 : X_2 : X_3)$ and $Q(Y_0 : Y_1 : Y_2 : Y_3)$ be rational points on a μ_4-Koblitz curve. Let $U_{ij} = X_iY_j$ in the following text. Point addition $P+Q$ on a μ_4-Koblitz curve can be calculated as

$$\big((U_{13} + U_{31})^2 : U_{02}U_{31} + U_{20}U_{13} + aF : (U_{02} + U_{20})^2 : U_{02}U_{13} + U_{20}U_{31} + aF\big)$$

where $F = (X_1 + X_3)(Y_1 + Y_3)(U_{02} + U_{20})$. It also can be calculated as

$$\big((U_{00} + U_{22})^2 : U_{00}U_{11} + U_{22}U_{33} + aG : (U_{11} + U_{33})^2 : U_{00}U_{33} + U_{11}U_{22} + aG\big)$$

where $G = (X_1 + X_3)(Y_1 + Y_3)(U_{00} + U_{22})$. This point addition requires $9\mathbf{M}+2\mathbf{S}$ and mixed addition requires $8\mathbf{M}+2\mathbf{S}$. The point addition with $a = 0$ is shown in Lemma 1 and that with $a = 1$ is shown in Lemma 2.

Lemma 1 (Corollary 5 in [14]) *Let $P(X_0 : X_1 : X_2 : X_3)$ and $Q(Y_0 : Y_1 : Y_2 : Y_3)$ be rational points on a μ_4-Koblitz curve with $a = 0$. Point addition $P + Q$ can be computed at the cost of* 7**M**+2**S** *as*

$$\begin{aligned}&\big((U_{00} + U_{22})^2 : U_{00}U_{11} + U_{22}U_{33} : (U_{11} + U_{33})^2 :\\ &(U_{00} + U_{22})(U_{11} + U_{33}) + U_{00}U_{11} + U_{22}U_{33}\big)\,.\end{aligned}$$

Mixed addition costs 6**M**+2**S**. *Point addition with both affine points costs* 5**M**+2**S**.

Lemma 2 (Theorem 1 in [18]) *Let $P(X_0 : X_1 : X_2 : X_3)$ and $Q(Y_0 : Y_1 : Y_2 : Y_3)$ be rational points on a μ_4-Koblitz curve with $a = 1$. Point addition $P + Q$ can be computed at the cost of* 8**M**+2**S** *as*

$$\begin{aligned}&\big((U_{00} + U_{22})^2 : U_{00}(U_{11} + H) + U_{22}(U_{33} + H) : (U_{11} + U_{33})^2 :\\ &(U_{00} + U_{22})(U_{11} + U_{33}) + U_{00}(U_{11} + H) + U_{22}(U_{33} + H)\big)\,,\end{aligned}$$

where $H = (X_1 + X_3)(Y_1 + Y_3)$. Mixed addition costs 7**M**+2**S**. *Point addition with both affine points costs* 6**M**+2**S**.

Jarvinen, Forsten, and Skytta first proposed $P \pm Q$ to improve the efficiency of scalar multiplication on Koblitz curves in affine coordinates [11]. Longa and Gebotys used $P \pm Q$ to improve the efficiency of pre-computation on elliptic curves over a prime field [19]. To avoid the expensive inversion, we will show the formulas of $P \pm Q$ on μ_4-Koblitz curves in Theorem 1. Avanzi, Dimitrov, Doche, and Sica [1] first introduced $\bar{\tau}$ to improve the efficiency of scalar multiplication. They noticed that $2 = \tau\bar{\tau}$ and computed $\bar{\tau}P$ requiring a point doubling and three square roots. Doche, Kohel, and Sica [8] proposed a new way to compute $\bar{\tau}P$ which induces a speedup on the scalar multiplication using double-base representation over 15% in LD coordinates. Inspired by their works, we introduce a new radix $\mu\bar{\tau}$ to speed up the pre-computation stage of scalar multiplication using window τNAF shown in Theorem 1.

Theorem 1 *Let $P(X_0 : X_1 : X_2 : X_3)$ and $Q(Y_0 : Y_1 : Y_2 : Y_3)$ be rational points on a μ_4-Koblitz curve. The two operations of $P+Q$ and $P-Q$ (($P \pm Q$)-operation) can be computed at the total cost of* 10**M**+3**S** *($a = 0$) and* 11**M**+3**S** *($a = 1$) when $X_2 = 1$, and $\mu\bar{\tau}P$ are calculated at the cost of* 2**M**+2**S**.

Proof. Let $P(X_0 : X_1 : X_2 : X_3)$, $Q(Y_0 : Y_1 : Y_2 : Y_3)$, and $-Q(Y_0 : Y_3 : Y_2 : Y_1)$.
When $a = 0$, $P + Q$ and $P - Q$ are computed as

$$\begin{aligned}P + Q &= \big((U_{00} + U_{22})^2 : U_{00}U_{11} + U_{22}U_{33} : (U_{11} + U_{33})^2 :\\ &\qquad (U_{00} + U_{22})(U_{11} + U_{33}) + U_{00}U_{11} + U_{22}U_{33}\big)\,,\\ P - Q &= \big((U_{11} + U_{33})^2 : U_{02}U_{33} + U_{20}U_{11} : (U_{02} + U_{20})^2 :\\ &\qquad (U_{02} + U_{20})(U_{11} + U_{33}) + U_{02}U_{33} + U_{20}U_{11}\big)\,.\end{aligned} \tag{2}$$

Notice that $U_{22} = Y_2$ and $U_{20} = Y_0$, the total cost of computing $P \pm Q$ is 10**M**+3**S**.

When $a = 1$, $P + Q$ and $P - Q$ are computed as

$$\begin{aligned} P + Q = \big(&(U_{00} + U_{22})^2 : U_{00}(U_{11} + H) + U_{22}(U_{33} + H) : (U_{11} + U_{33})^2 : \\ &(U_{00} + U_{22})(U_{11} + U_{33}) + U_{00}(U_{11} + H) + U_{22}(U_{33} + H)\big)\,, \\ P - Q = \big(&(U_{11} + U_{33})^2 : U_{02}(U_{33} + H) + U_{20}(U_{11} + H) : (U_{02} + U_{20})^2 : \\ &(U_{02} + U_{20})(U_{11} + U_{33}) + U_{02}(U_{33} + H) + U_{20}(U_{11} + H)\big)\,, \end{aligned} \tag{3}$$

where $H = (X_1 + X_3)(Y_1 + Y_3)$. Since $U_{22} = Y_2$ and $U_{20} = Y_0$, the total cost of computing $P \pm Q$ is 11**M**+3**S**.

Notice that $2 = \tau\bar{\tau}$. We have $2\mu P = \tau(\mu\bar{\tau}P)$.

It is pointed out that there is one typographical error in Section 6 of [15], the correct doubling formulas are in Kohel's slides [16] where $2\mu P$ is computed as

$$\big((X_0 + X_2)^4 : (X_0X_3 + X_1X_2)^2 : (X_1 + X_3)^4 : (X_0X_1 + X_2X_3)^2\big)\,.$$

Then

$$\mu\bar{\tau}P = \big((X_0 + X_2)^2 : (X_0X_3 + X_1X_2) : (X_1 + X_3)^2 : (X_0X_1 + X_2X_3)\big)\,. \tag{4}$$

When $a = 0$, since $(X_0X_3 + X_1X_2) = (X_0 + X_1)(X_2 + X_3) + (X_0 + X_2)^2 + (X_1 + X_3)^2$ and $(X_0X_1 + X_2X_3) = (X_0 + X_2)(X_1 + X_3) + (X_0X_3 + X_1X_2)$, the cost of $\mu\bar{\tau}P$ is 2**M**+2**S**.

When $a = 1$, since $(X_0X_3 + X_1X_2) = (X_0 + X_1)(X_2 + X_3) + (X_0 + X_2)^2$ and $(X_0X_1 + X_2X_3) = (X_0 + X_2)(X_1 + X_3) + (X_0X_3 + X_1X_2)$, the cost of $\mu\bar{\tau}P$ is 2**M**+2**S**.

□

Since separate computations of $P + Q$ and $P - Q$ require 12**M**+4**S** ($a = 0$) and 14**M**+4**S** ($a = 1$), our formulas save 2**M**+**S** ($a = 0$) and 3**M**+**S** ($a = 1$). In the case of $a = 0$, using our formulas of $P \pm Q$, Solinas' pre-computation scheme saves 2**M**+**S** for $w = 4$ and 4**M**+2**S** for $w = 5$; Hankerson, Menezes, and Vanstone's pre-computation scheme saves 2**M**+**S** for $w = 4$, 4**M**+2**S** for $w = 5$, and 10**M**+5**S** for $w = 6$; Trost and Xu's pre-computation scheme saves 4**M**+2**S** for $w = 6$.

Our formulas of $\mu\bar{\tau}$-operation save 4**M**($a = 0$) and 5**M**($a = 1$). The costs of point operations including $(P \pm Q)$-operation and $\mu\bar{\tau}P$ are summarized in Table 6. Notice that formulas of $(P \pm Q)$-operation are the two forms of the formulas of point addition and formulas of $\mu\bar{\tau}P$ are part of the formulas of point doubling. This leads to software and hardware implementations with simplicity. These new efficient point operations will be used to improve the arithmetics on a μ_4-Koblitz curve.

4 A Novel Pre-Computation Scheme

Solinas' pre-computation in Section 7.4 of [27], Hankerson, Menezes, and Vanstone's pre-computation shown as Tables 3.9 and 3.10 in [10], and Trost and

Table 6. Costs of point operations on a μ_4-Koblitz curve

Point operation	cost ($a=0$)	cost ($a=1$)
($P \pm Q$)-operation (this work)	10**M**+3**S** (Equation (2))	11**M**+3**S** (Equation (3))
$\mu\bar{\tau}P$ (this work)	2**M**+2**S** (Equation (4))	2**M**+2**S** (Equation (4))

Xu's pre-computation shown as Tables 5 and 6 in [28] all have a pre-computation scheme on E_0 and another pre-computation scheme on E_1. In this section, we will introduce a unified pre-computation without treating $a=0$ and $a=1$ separately. Our method is to write pre-computations with variable curve coefficient hidden in μ. Let $c_i \in R_i$ and $c_i = g + h\mu\tau$ for $i \in I_w$. Then $Q_i = c_i P$ works on both E_0 and E_1. We call $Q_i = c_i P$ a unified pre-computation scheme when c_i has the form $g + h\mu\tau$ for all $i \in I_w$. Trost and Xu's pre-computation can be unified. Take $w=4$ for example, we have $Q_5 = -P + \mu\tau P$, $Q_7 = P + \mu\tau P$, $Q_3 = -3P + \mu\tau P$. Also Solinas' pre-computation, and Hankerson, Menezes, and Vanstone's pre-computation can be unified.

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

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 τ^w whose norms are smaller than 2^w for each $i \in I_w$. Since elements of I_w are odd integers, we will 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 3 *We have the following facts:*

1. *If $g + h\tau \in R_i$ for some $i \in I_w$, then $g - h\tau \notin R_i$ for any $h \neq 0$.*
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 [28], we know that if $g + h\tau \in R_i$, then $|g| < \frac{2^{\frac{w+2}{2}}}{\sqrt{3}}$ and $|h| < 2^{\frac{w}{2}}$.

(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 $|g|, |g'| < \frac{2^{\frac{w+2}{2}}}{\sqrt{3}}$, then $|g-g'| < 2 \cdot \frac{2^{\frac{w+2}{2}}}{\sqrt{3}} \leq 2^w$. 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 4 *Let $i \in I_w$, then $\#R_i \leq \left\lfloor 2^{\frac{w+2}{2}} \right\rfloor$.*

Proof. If $g + h\tau \in R_i$, then $|h| < 2^{\frac{w}{2}}$. So the cardinality of $T = \{h \in \mathbb{Z} | g + h\tau \in R_i$ for some odd number $g\}$ is less than $2 \cdot 2^{\frac{w}{2}}$. By Lemma 3, for each $h \in T$, there is only one g available such that $g + h\tau \in R_i$. Thus $\#R_i = \#T \leq \left\lfloor 2^{\frac{w+2}{2}} \right\rfloor$. □

If $g + h_1\tau \equiv g + h_2\tau \pmod{\tau^w}$, then $s_w(h_2 - h_1) \equiv 0 \pmod{2^w}$. Since s_w is even and $s_w/2$ is odd, $h_2 = h_1 + c \cdot 2^{w-1}$. 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}, $i \in I_w$ when g is odd. On average, $\#R_i$ is less than 4.62. We have calculated out that $\#R_i \leq 3$ for $i \in I_w$ and $3 \leq w \leq 10$.

4.2 Calculating R_i

We propose a plane search to generate R_i, $i \in I_w$, shown as Algorithm 2. 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 treat it as the point (g, h) on the Euclidean plane. To determine whether $g + h\mu\tau$ is in the set R_i for some i satisfying $2^w | g - i + h\mu s_w$, we search all points (g, h) and append $g + h\mu\tau$ to the corresponding R_i where $-\left\lfloor \frac{2^{\frac{w+2}{2}}}{\sqrt{3}} \right\rfloor \leq g \leq \left\lfloor \frac{2^{\frac{w+2}{2}}}{\sqrt{3}} \right\rfloor$, $-\left\lfloor 2^{\frac{w}{2}} \right\rfloor \leq h \leq \left\lfloor 2^{\frac{w}{2}} \right\rfloor$, and g is odd. We collect all such elements and form a set $C = \{c_i | c_i \in R_i,\ i \in I_w\}$. Then $Q_i = c_i P$ with $c_i \in C$ for all $i \in I_w$ form a unified pre-computation. We set the trivial case $c_1 = 1$.

Algorithm 2 Plane search to generate R_i, $i \in I_w$

Computation

1. $R_i \leftarrow <>$
2. for g from $-\left\lfloor \frac{2^{\frac{w+2}{2}}}{\sqrt{3}} \right\rfloor$ to $\left\lfloor \frac{2^{\frac{w+2}{2}}}{\sqrt{3}} \right\rfloor$ and g is odd
   for h from $-\left\lfloor 2^{\frac{w}{2}} \right\rfloor$ to $\left\lfloor 2^{\frac{w}{2}} \right\rfloor$
   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

4.3 Our Novel Pre-Computation

We design a novel pre-computation for window τNAF with widths from 4 to 8.

Theorem 2 *Let $P = (x_P, \lambda_P)$ and $Q_i = (X_i, \Lambda_i, Z_i)$ with $i \in I_w$. There exists a unified pre-computation scheme shown in Tables* 7, 14, *and* 15 *requiring* 6**M**+6**S**, 18**M**+17**S**, 44**M**+32**S**, 88**M**+62**S**, *and* 186**M**+123**S** *on a μ_4-Koblitz curve with $a = 0$ and* 6**M**+6**S**, 19**M**+17**S**, 47**M**+32**S**, 93**M**+72**S**, *and* 198**M**+123**S** *with $a = 1$ for window τNAF with widths from* 4 *to* 8 *respectively.*

Table 7. Novel pre-computation for widths from 4 to 6

	c_i		Q_i	$a = 0/a = 1$
$w = 4$				**6M**+6**S**
	$c_5 = -1 + \mu\tau$	$c_5 = -\mu\bar{\tau}$	$Q_5 = -\mu\bar{\tau}P$	2**M**+2**S**
	$c_7 = 1 + \mu\tau$	$c_7 = \mu\bar{\tau}c_5$	$Q_7 = -(\mu\bar{\tau})^2P$	2**M**+2**S**
	$c_3 = -3 + \mu\tau$	$c_3 = -\mu\bar{\tau}c_7$	$Q_3 = (\mu\bar{\tau})^3P$	2**M**+2**S**
$w = 5$				18**M**+17**S**/19**M**+17**S**
	$c_5 = -1 + \mu\tau$	$c_5 = -\mu\bar{\tau}$	$Q_5 = -\mu\bar{\tau}P$	2**M**+2**S**
	$c_7 = 1 + \mu\tau$	$c_7 = \mu\bar{\tau}c_5$	$Q_7 = -(\mu\bar{\tau})^2P$	2**M**+2**S**
	$c_3 = -3 + \mu\tau$	$c_3 = -\mu\bar{\tau}c_7$	$Q_3 = (\mu\bar{\tau})^3P$	2**M**+2**S**
	$c_{15} = 1 - 3\mu\tau$	$c_{15} = -\mu\bar{\tau}c_3$	$Q_{15} = -(\mu\bar{\tau})^4P$	2**M**+2**S**
	$c_{11} = -1 + 2\mu\tau$	$c_{11} = \mu\tau + c_5$	$Q_{11} = \mu\tau P + Q_5$	(6**M**+2**S**)+3**S**/(7**M**+2**S**) + 3**S***
	$c_9 = 3 + \mu\tau$	$c_9 = \mu\bar{\tau}c_{11}$	$Q_9 = \mu\bar{\tau}Q_{11}$	2**M**+2**S**
	$c_{13} = -5 + 3\mu\tau$	$c_{13} = -\mu\bar{\tau}c_9$	$Q_{13} = -(\mu\bar{\tau})^2Q_{11}$	2**M**+2**S**
$w = 6$				44**M**+32**S**/47**M**+32**S**
	$c_{27} = 1 - \mu\tau$	$c_{27} = \mu\bar{\tau}$	$Q_{27} = \mu\bar{\tau}P$	2**M**+2**S**
	$c_{25} = -1 - \mu\tau$	$c_{25} = \mu\bar{\tau}c_{27}$	$Q_{25} = (\mu\bar{\tau})^2P$	2**M**+2**S**
	$c_{29} = 3 - \mu\tau$	$c_{29} = -\mu\bar{\tau}c_{25}$	$Q_{29} = -(\mu\bar{\tau})^3P$	2**M**+2**S**
	$c_{15} = 1 - 3\mu\tau$	$c_{15} = \mu\bar{\tau}c_{29}$	$Q_{15} = -(\mu\bar{\tau})^4P$	2**M**+2**S**
	$c_{21} = -5 - \mu\tau$	$c_{21} = \mu\bar{\tau}c_{15}$	$Q_{21} = -(\mu\bar{\tau})^5P$	2**M**+2**S**
	$c_3 = 3$	$c_3 = \mu\tau + c_{29}$	$Q_3 = \mu\tau P + Q_{29}$	
	$c_9 = -3 + 2\mu\tau$	$c_9 = \mu\tau - c_{29}$	$Q_9 = \mu\tau P - Q_{29}$	(10**M**+3**S**)+3**S**/(11**M**+3**S**)+3**S***
	$c_{13} = -1 - 3\mu\tau$	$c_{13} = -\mu\bar{\tau}c_9$	$Q_{13} = -(\mu\bar{\tau})Q_9$	2**M**+2**S**
	$c_{31} = -7 + \mu\tau$	$c_{31} = \mu\bar{\tau}c_{13}$	$Q_{31} = -(\mu\bar{\tau})^2Q_9$	2**M**+2**S**
	$c_{17} = 3 - 3\mu\tau$	$c_{17} = \mu\bar{\tau}c_3$	$Q_{17} = \mu\bar{\tau}Q_3$	2**M**+2**S**
	$c_{11} = -3 - 3\mu\tau$	$c_{11} = \mu\bar{\tau}c_{17}$	$Q_{11} = (\mu\bar{\tau})^2Q_3$	2**M**+2**S**
	$c_{23} = -1 + 4\mu\tau$	$c_{23} = \mu\tau - c_{15}$	$Q_{23} = \mu\tau P - Q_{15}$	6**M**+2**S**/7**M**+2**S**
	$c_{19} = -7 - \mu\tau$	$c_{19} = -\mu\bar{\tau}c_{23}$	$Q_{19} = -\mu\bar{\tau}Q_{23}$	2**M**+2**S**
	$c_5 = 5$	$c_5 = -\mu\tau - c_{21}$	$Q_5 = -\mu\tau P - Q_{21}$	6**M**+2**S**/7**M**+2**S**
	$c_7 = 5 - 5\mu\tau$	$c_7 = \mu\bar{\tau}c_5$	$Q_7 = \mu\bar{\tau}Q_5$	2**M**+2**S**

* "+3**S**" is the cost of τP. For window width 6, Q_3 and Q_9 can be computed as one $(P \pm Q)$-operation.

Proof. The explicit design of calculating pre-computations for window τNAF with widths from 4 to 6 is shown as Table 7, for that with width 7 is shown as Table 14 in Appendix A.1, and for that with width 8 is shown as Table 15 in Appendix A.2. Let $c_i = g + h\mu\tau$ for each $i \in I_w$ in Tables 7, 14, and 15. Since $c_i = g + h\mu\tau$ for each $i \in I_w$, our pre-computation scheme for w from 4 to 8 is unified. Since $g + h\mu s_w \equiv i \pmod{2^w}$ and $N(c_i) < 2^w$ for all $i \in I_w$, this novel pre-computation is correct for window τNAF with widths from 4 to 8.

We show our novel pre-computation for window τNAF with widths 4, 5, and 6 as follows.

1. $w = 4$. $Q_5 = -(\mu\bar{\tau}P)$, $Q_7 = -(\mu\bar{\tau})^2P$, $Q_3 = (\mu\bar{\tau})^3P$ are shown as Table 7. Our pre-computation scheme for window τNAF with width 4 requires 6**M**+6**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 Table 7. This pre-computation scheme requires 18**M**+17**S** with $a = 0$ and 19**M**+17**S** with $a = 1$.
3. $w = 6$. Let $\tau P = (x_{\tau P}, \lambda_{\tau P}) = (x_P^2, \lambda_P^2)$. $Q_{27} = \mu\bar{\tau}P$, $Q_{25} = (\mu\bar{\tau})^2P$, $Q_{29} = -(\mu\bar{\tau})^3P$, $Q_{15} = -(\mu\bar{\tau})^4P$, $Q_{21} = -(\mu\bar{\tau})^5P$, $(Q_3, Q_9) = \mu\tau P \pm Q_{29}$ ($Q_3 = \mu\tau P + Q_{29}$, $Q_9 = \mu\tau P - Q_{29}$), $Q_{13} = -(\mu\bar{\tau})Q_9$, $Q_{31} = -(\mu\bar{\tau})^2Q_9$, $Q_{17} = \mu\bar{\tau}Q_3$, $Q_{11} = (\mu\bar{\tau})^2Q_3$, $Q_{23} = \mu\tau P - Q_{15}$, $Q_{19} = -\mu\bar{\tau}Q_{23}$, $Q_5 = -\mu\tau P - Q_{21}$,

$Q_7 = \mu\bar{\tau}Q_5$ are shown as Table 7. This scheme requires 44**M**+32**S** with $a = 0$ and 47**M**+32**S** with $a = 1$.

The explicit computing process and the value of c_i for window τNAF with widths from 4 to 6 are shown as Table 7; those for window τNAF with width 7 are shown as Table 14; and those for window τNAF with width 8 are shown as Table 15. □

For each Q_i ($i = 3, 5, \ldots, 2^{w-1} - 1$), one point addition is necessary. We employ $\mu\bar{\tau}(P)$ and $(P \pm Q)$-operations to replace point addition which leads to a speedup of our pre-computation algorithm. Next, we will compare our scheme with other pre-computation schemes.

4.4 Comparison of Pre-Computation Schemes in M and S

The ratio of **I**/**M** and that of **S**/**M** both affect the cost of pre-computation schemes and that of scalar multiplications. Suppose that **I**/**M**=10, **S**/**M**=0; or **I**/**M**=10, **S**/**M**=0.2; or **I**/**M**=150, **S**/**M**= 0.5. The first two cases are both suggested by Bernstein and Lange in their explicit-formulas database [4]. The third case suits for binary fields over desktop architectures embedded with the carry-less multiplication instruction [9]. The first two ratios are reasonable in the experiments of our environments shown as Section 6 where **I**/**M**=10 and 0.06<**S**/**M**<0.12.

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 on the μ_4-Koblitz curves with $a = 0$ and $a = 1$ for window τNAF are summarized in Table 1. Our pre-computation scheme is the fastest one among these four pre-computation schemes. Our novel pre-computation scheme is about two times faster than Trost and Xu's scheme for window τNAF with widths 4, 5, and 6 for all three cases.

5 Scalar Multiplications Using Window τNAF on μ_4-Koblitz Curves

Let the costs of pre-computation schemes for window τNAF with width w be denoted by Pre_w.

5.1 Expected Costs of Scalar Multiplications

Scalar multiplication using window τNAF has two situations.

1. Scalar multiplication uses pre-computations in projective coordinates. It requires m τ-operations, $\frac{m}{w+1} \cdot \frac{2^{w-2}-1}{2^{w-2}}$ point additions, $\frac{m}{w+1} \cdot \frac{1}{2^{w-2}}$ mixed additions, and the pre-computation. Scalar multiplication is expected to cost

$$4m\mathbf{S} + \frac{m}{w+1}\left((7+a)\mathbf{M} + 2\mathbf{S} - \frac{1}{2^{w-2}}\mathbf{M}\right) + \mathrm{Pre}_w.$$

2. Scalar multiplication uses pre-computations in affine coordinates. This method fully uses mixed additions and requires Montgomery trick to translate the pre-computation points in projective coordinates to those in affine coordinates. It requires m τ-projective operations, $\frac{m}{w+1}$ mixed additions, Montgomery trick, and the pre-computation. Scalar multiplication is expected to cost

$$4m\mathbf{S} + \frac{m}{w+1}((6+a)\mathbf{M} + 2\mathbf{S}) + \mathbf{I} + (6 \cdot 2^{w-2} - 9)\mathbf{M} + \mathrm{Pre}_w.$$

For window τNAF with width w, one should choose Case 1 or Case 2 to compute the scalar multiplication. The selection is not affected by the efficiency of the pre-computation. For the case of $a = 0$, the lowest costs of scalar multiplications on K-233, K-283, K-409, and K-571 using μ_4-Koblitz curves utilizing our pre-computation scheme and Trost and Xu's pre-computation scheme are summarized in Table 8. For the case of $a = 1$, the lowest costs of scalar multiplications on K1-163, K1-283, K1-359, and K1-701 utilizing our pre-computation scheme and Trost and Xu's pre-computation scheme are summarized in Table 9.

Table 8. The expected costs of scalar multiplications on K-233, K-283, K-409, and K-571 using μ_4-Koblitz curves in **M**

		K-233(w)	K-283(w)	K-409(w)	K-571(w)
	τNAF	466	566	818	1142
S=0**M**	Trost, Xu	306.0(5)	363.3(5)	492.3(6)	652.9(6)
	Ours	274.9(6)	324.5(6)	444.3(7)	585.4(7)
	regular τNAF	683.5	830.1	1199.7	1674.9
S=0.2**M**	Trost, Xu	511.9(5)	612.5(5)	850.1(6)	1149.5(6)
	Ours	481(6)	573.4(6)	804.3(7)	1083.1(7)
	regular τNAF	1009.7	1226.3	1772.3	2474.3
S=0.5**M**	Trost, Xu	835.2(6)	991.9(6)	1386.8(6)	1894.5(6)
	Ours	790.2(6)	946.9(6)	1341.8(6)	1829.8(7)

Table 9. The expected costs of scalar multiplications on K1-163, K1-283, K1-359, and K1-701 using μ_4-Koblitz curves in **M**

		K1-163(w)	K1-283(w)	K1-359(w)	K1-701(w)
	τNAF	380.3	660.3	837.7	1635.7
S=0**M**	Trost, Xu	259.9(5)	417.4(5)	509.1(6)	896.9(6)
	Ours	231.8(6)	367.9(6)	450.6(7)	791.3(7)
	τNAF	532.5	924.5	1172.7	2289.9
S=0.2**M**	Trost, Xu	405.2(5)	666.7(5)	824(6)	1505(6)
	Ours	377.9(6)	616.9(6)	768.1(7)	1399.5(7)
	τNAF	760.7	1320.7	1675.3	3271.3
S=0.5**M**	Trost, Xu	623.1(5)	1040.6(5)	1296.4(6)	2417.3(6)
	Ours	594.6(5)	990.3(6)	1239.4(6)	2311.9(7)

5.2 Expected Costs of Constant-Time Scalar Multiplications

When a constant running time is required, a regular window τNAF [23], the improved recoding of zero-free representation [22, 29], is used to implement scalar multiplication. Scalar multiplication using pre-computations in projective coordinates requires

$$4m\mathbf{S} + \frac{m}{w-1}((7+a)\mathbf{M} + 2\mathbf{S}) + \mathrm{Pre}_w.$$

Scalar multiplication using pre-computations in affine coordinates requires

$$4m\mathbf{S} + \frac{m}{w-1}((6+a)\mathbf{M} + 2\mathbf{S}) + \mathbf{I} + (6 \cdot 2^{w-2} - 9)\mathbf{M} + \mathrm{Pre}_w.$$

We summarize the lowest costs of constant-time scalar multiplications using our pre-computation scheme and Trost and Xu's pre-computation scheme on curves with $a = 0$ in Table 10 and on curves with $a = 1$ in Table 11. Our pre-computation saves 9**M**+6**S** with $a = 0$ and 12**M**+6**S** with $a = 1$ for $w = 4$, 21**M**+3**S** with $a = 0$ and 27**M**+3**S** with $a = 1$ for $w = 5$, 43**M**+4**S** with $a = 0$ and 55**M**+4**S** with $a = 1$ for $w = 6$, compared to the state-of-the-art pre-computation. Our pre-computation scheme only requires 88**M**+62**S** with $a = 0$ and 93**M**+62**S** with $a = 1$ for $w = 7$, and 186**M**+123**S** with $a = 0$ and 198**M**+123**S** with $a = 1$ for $w = 8$. Since constant-time scalar multiplication usually uses window τNAF with a bigger window width, the ratios of the improvements of scalar multiplication become higher.

Table 10. The expected costs of constant-time scalar multiplications on K-233, K-283, K-409, and K-571 using μ_4-Koblitz curves in **M**

		K-233(w)	K-283(w)	K-409(w)	K-571(w)
	regular τNAF	1398	1698	2454	3426
S=0**M**	Trost, Xu	413.2(6)	483.2(6)	659.6(6)	869.2(6,M)
	Ours	359.8(7)	418.2(7)	565.2(7)	754.2(7)
	regular τNAF	1677.6	2037.6	2944.8	4111.2
S=0.2**M**	Trost, Xu	625.4(6)	739.4(6)	1026.7(6)	1378.9(6,M)
	Ours	574.2(7)	675.8(7)	932(7)	1261.4(7)
	regular τNAF	2097	2547	3681	5139
S=0.5**M**	Trost, Xu	943.8(6)	1123.8(6)	1577.4(6)	2160.6(6)
	Ours	895.7(7)	1062.3(7)	1482.3(7)	2022.3(7)

If we use Montgomery trick, we denote it by M. This notation is also used in the following tables.

Table 11. The expected costs of constant-time scalar multiplications on K1-163, K1-283, K1-359, and K1-701 using μ_4-Koblitz curves in **M**

		K1-163(w)	K1-283(w)	K1-359(w)	K1-701(w)
	regular τNAF	1141	1981	2513	4907
S=0**M**	Trost, Xu	362.8(6)	554.8(6)	676.4(6)	1180.4(6,M)
	Ours	307.8(6)	470.3(7)	571.7(7)	999.1(8)
	regular τNAF	1336.6	2320.6	2943.8	5748.2
S=0.2**M**	Trost, Xu	513.4(6)	811(6)	999.5(6)	1804.5(6,M)
	Ours	457.6(6)	728(7)	895.2(7)	1624.6(8)
	regular τNAF	1630	2830	3590	7010
S=0.5**M**	Trost, Xu	739.4(6)	1195.4(6)	1484.2(6)	2783.8(6)
	Ours	682.4(6)	1114.5(7)	1380.5(7)	2562.8(8)

6 Experiments

Miracl lib [26] is used to implement field arithmetics over $\mathbb{F}_{2^m}$. Our experiments are tested by C++ programs compiled by Microsoft visual studio 2015. The processor is Intel® Core^{TM} i7-6567U 3.3 GHZ with Skylake architecture and the operating system is 64-bit Windows 10.

6.1 Pre-Computation Schemes on μ_4-Koblitz Curves

We run each pre-computation scheme 1000 times on six Koblitz curves. The time costs of pre-computation schemes on K1-163, K-233, K-283, K1-283, K-409, and K-571 using μ_4-Koblitz curves for window τNAF with widths from 4 to 6 are shown in Table 12.

Table 12. Time costs of pre-computations on K1-163, K-233, K-283, K1-283, K-409, and K-571 using μ_4-Koblitz curves in μs

		K1-163	K-233	K-283	K1-283	K-409	K-571
$w=4$	Solinas	4.4	5.36	7.08	8.36	10.48	12.35
	Hankerson, Menezes, Vanstone	4.4	5.36	7.08	8.36	10.48	12.35
	Trost, Xu	4.36	5.28	6.52	7.64	10	11.76
	Ours	1.76	2.24	3.04	3.4	4.5	5.432
$w=5$	Solinas	11.24	13.68	17.6	20.72	27.86	31.81
	Hankerson, Menezes, Vanstone	11.52	14.04	18.32	20.92	29.12	33.77
	Trost, Xu	10.88	13.28	17.56	20.36	27.47	31.75
	Ours	4.96	6.44	8.36	9.16	13.5	15.2
$w=6$	Hankerson, Menezes, Vanstone	25.16	30.68	40.48	46.36	63.83	73.64
	Trost, Xu	24.88	30.36	39.24	45.28	62.96	71.89
	Ours	11.44	15.32	19.96	21.16	31.72	36.54

Our pre-computation scheme is about two times faster than Trost and Xu's scheme. Within the bounds of the error, the practical implementations are consistent with the theoretical analysis. The reason of some tiny differences is that a few field additions were ignored, that the number of temporary variables affects the performance, and that the ratio of $\mathbf{S}/\mathbf{M}$ is about 0.06 to 0.12 which depends on the size of the binary field.

6.2 Scalar Multiplications on μ_4-Koblitz Curves

The costs of constant-time scalar multiplications on K1-163, K-233, K-283, K1-283, K-409, and K-571 using μ_4-Koblitz curves are shown in Table 13. Our constant-time scalar multiplication is over 3 times faster, compared to the state-of-the-art non-pre-computation-based constant-time scalar multiplication. The constant-time scalar multiplication using our pre-computation on μ_4-Koblitz curves runs in 85.6%, 88.7%, 87.9%, 85.2%, 87.7%, and 87.9% the time of that using Trost and Xu's pre-computation on μ_4-Koblitz curves. The experimental results also show that the lowest constant-time scalar multiplication using our pre-computation usually employs width 7, and that using Trost and Xu's pre-computation usually employs width 6.

Table 13. Time cost of scalar multiplications using μ_4-Koblitz curves in μs

		K1-163(w)	K-233(w)	K-283(w)	K1-283(w)	K-409(w)	K-571(w)
	τNAF	70.42	98.6	171.9	167.3	384.2	424.6
	Trost, Xu	48.9(5)	70.23(5)	114.9(5)	132.1(5)	225(6)	268.4(6)
	Ours	44.75(6)	64.05(6)	104.3(6)	117.8(6)	207.4(7)	243.3(7)
constant-time	regular τNAF	173.7	265.6	432.4	491.8	860.1	1038.5
	Trost, Xu	63.95(6)	88.7(6)	143.6(6)	164.8(6)	283.6(6)	336.2(6,M)
	Ours	54.77(6)	78.67(7)	126.2(7)	140.5(7)	248.8(7)	294.7(7)

7 Conclusion

In the previous works of scalar multiplication using window τNAF [10,22,23,27–29], the authors employed a window τNAF with width at most 6. From Tables 8, 9, 10, 11, and 13, scalar multiplication using our pre-computation usually employs a bigger window width (e.g., 7) to achieve a lower cost of the total scalar multiplication.

In Appendix B, we employed our pre-computation scheme on Koblitz curves using LD coordinates. Our pre-computation scheme requires 5**M**+6**S**, 19**M**+19**S**, 51**M**+40**S**, 99**M**+76**S**, and 214**M**+158**S** when $a = 0$, and 5**M**+3**S**, 19**M**+13**S**, 51**M**+29**S**, 99**M**+53**S**, and 214**M**+113**S** when $a = 1$ using LD coordinates for window τNAF with widths from 4 to 8 respectively. Constant-time scalar multiplication using Trost and Xu's pre-computation requires 74.35, 109.4, 189.8, 357.9, and 433.1 μs on K1-163, K-233, K-283/K1-283, K-409, and K-571 respectively. Non-pre-computation-based constant-time scalar multiplication 216.3, 339.7, 547.8, 1078.3, and 1330.6 μs on these curves. These experimental results show that constant-time scalar multiplication using our pre-computation on μ_4-Koblitz curves runs in 73.7%, 71.9%, 66.5%, 74%, 69.5%, and 68% the time of Trost and Xu's work on K1-163, K-233, K-283, K1-283, K-409, and K-571 respectively where they used LD coordinates to perform scalar multiplication. Our scalar multiplication on μ_4-Koblitz curves is about 4 times faster than non-pre-computation-based constant-time scalar multiplication in LD coordinates and saves up to 33.5% on the scalar multiplication compared to scalar multiplication using Trost and Xu's pre-computation in LD coordinates.

In Appendix C, we employed our pre-computation scheme on Koblitz curves using λ-coordinates. The costs of our pre-computation scheme are 7**M**+5**S**, 26**M**+16**S**, 66**M**+36**S**, 135**M**+72**S**, and 282**M**+148**S** using λ-projective coordinates for window τNAF with widths from 4 to 8 respectively. Constant-time scalar multiplication using Trost and Xu's pre-computation requires 71.21, 102.2, 176.7, 335.9, and 402.5 μs on K1-163, K-233, K-283/K1-283, K-409, and K-571 respectively. Non-pre-computation-based constant-time scalar multiplication 211.7, 332.3, 540.8, 1065.2, and 1316.1 μs on these curves. These experimental results show that constant-time scalar multiplication using our pre-computation on μ_4-Koblitz curves runs in 76.9%, 77%, 71.4%, 79.5%, 74.1%, and 73.2% the time of Trost and Xu's work on K1-163, K-233, K-283, K1-283, K-409, and K-571 respectively where they used λ-coordinates to perform scalar multiplication. Based on our novel pre-computation, the efficient arithmetics on μ_4-Koblitz curves, and a bigger window width, our scalar multiplication on μ_4-Koblitz curves is about 4 times faster than non-pre-computation-based constant-time scalar multiplication in λ-coordinates and can save up to 28.6% on the scalar multiplication compared to [28].

It is noted that the arithmetic of Koblitz curves has been of theoretical and practical importance since the start of elliptic curve cryptography. Our results make a significant progress on the scalar multiplication for Koblitz curves which is a long-standing and well-studied area.

The idea of using $\mu\bar{\tau}$ to design an efficient pre-computation scheme and using a window τNAF with a bigger window width to improve the efficiency of scalar multiplication can be extended to Koblitz curves over $\mathbb{F}_{3^m}$ and $\mathbb{F}_{q^m}$ for some small primes $q \geq 5$. The efficient $\mu\bar{\tau}$-operations can also be used to speed up scalar multiplication utilizing double-base chain [30] and double-base number system [1], and to speed up multi-scalar multiplication utilizing double-base number system [8].

Acknowledgments

The authors would like to thank the anonymous reviewers for many helpful comments and thank Bao Li, Kunpeng Wang, Xianhui Lu, and Song Tian for their helpful suggestions. This work is supported by the National Natural Science Foundation of China (No. 61872442 and U1936209), by National Key Research and Development Program of China (No. 2018YFA0704702), and by Department of Science and Technology of Shandong Province of China (No. 2019JZZY010133). W. Yu is supported by Beijing Municipal Science & Technology Commission (No. Z191100007119006), by the National Cryptography Development Fund (No. MMJJ20180216), and by the National Natural Science Foundation of China (No. 61772515 and 61502487).

References

1. Avanzi R.M., Dimitrov V.S., Doche C., Sica F.: Extending scalar multiplication using double bases. ASIACRYPT 2006. LNCS, vol. 4284, pp. 130-144. Springer, Heidelberg, 2006.
2. Barker E.: Draft NIST special publication 800-57 part 1 revision 5 – recommendation for key management, part 1: general. May 2020. https://doi.org/10.6028/NIST.SP.800-57pt1r5
3. Barker E., Chen L., Roginsky A., Vassilev A., Davis R.: NIST special publication 800-56A revision 3 – recommendation for pair-wise key-establishment schemes using discrete logarithm cryptography. April 2018. https://doi.org/10.6028/NIST.SP.800-56Ar3
4. Bernstein D.J., Lange T.: Explicit-formulas database, 2020, `http://hyperelliptic.org/EFD/`
5. Blake I., Murty V., Xu G.: A note on window τ-NAF algorithm, Inf. Process. Lett., vol. 95, no. 5, pp. 496-502, 2005.
6. Blake I., Murty V., Xu G.: Nonadjacent radix-τ expansions of integers in Euclidean imaginary quadratic number fields, Canadian J. Math., vol. 60, pp. 1267-1282, 2008.

7. Bos J., Lenstra A., Te Riele H., Shumow D.: Introduction. In Bos J. and Lenstra A.(Eds.), Topics in Computational Number Theory Inspired by Peter L. Montgomery (pp. 1-9). Cambridge: Cambridge University Press. doi:10.1017/9781316271575.002, October, 2017.
8. Doche C., Kohel D.R., Sica F.: Double Base Number System for multi scalar multiplications. EUROCRYPT 2009. LNCS, vol. 5479, pp. 502-517. Springer, Heidelberg, 2009.
9. Gueron S., Kounavis M.: Intel carry-less multiplication instruction and its usage for computing the GCM mode, Revision 2.02, Intel, April 2014. `https://software.intel.com/sites/default/files/managed/72/cc/clmul-wp-rev-2.02-2014-04-20.pdf`
10. Hankerson D., Menezes A., Vanstone S.: Guide to elliptic curve cryptography. New York, NY, USA: Springer-Verlag, 2004.
11. Jarvinen K., Forsten J., Skytta J.: FPGA Design of Self-certified Signature Verification on Koblitz Curves. CHES 2007, LNCS, vol. 4727, pp. 256-271. Springer, Heidelberg, 2007.
12. Koblitz N.: CM-curves with good cryptographic properties, in Proc. 11th Annu. Int. Cryptol. Conf. Adv. Cryptol., pp. 279-287, 1992.
13. Koblitz N.: p-adic numbers, p-adic analysis, and zeta-functions. New York, NY, USA: Springer, 1996.
14. Kohel D.: Efficient arithmetic on elliptic curves in characteristic 2, 2016. `https://arxiv.org/abs/1601.03669`
15. Kohel D.: Twisted μ_4-Normal Form for Elliptic Curves. EUROCRYPT 2017, Part I. LNCS, vol. 10210, pp. 659-678. Springer, Heidelberg, 2017.
16. Kohel D.: Twisted μ_4-normal form for elliptic curves. EUROCRYPT 2017, Paris, 2017. `https://eurocrypt.iacr.org/2017/slides/A03-twisted.pdf`
17. Lange, T.: A note on Lopez-Dahab coordinates. Cryptology ePrint Archive, Report 2004/323 (2004). https:https://eprint.iacr.org/2004/323.pdf
18. Li W., Yu W. Li B., Fan X.: Speeding up scalar multiplication on Koblitz curves using μ_4 coordinates. ACISP 2019, LNCS, vol. 11547, pp. 1-10. Springer, Heidelberg, 2019.
19. Longa P., Gebotys C.: Novel precomputation schemes for elliptic curve cryptosystems, ACNS 2009, LNCS, vol. 5536, pp. 71-88. Springer, Heidelberg, 2009.
20. López J., Dahab R.: Improved algorithms for elliptic curve arithmetic in $GF(2^n)$, in Proc. Selected Areas Cryptography, LNCS, vol. 1556, pp. 201-212, 1998.
21. National Institute of Standards and Technology(NIST).: Digital signature standard(DSS). FIPS PUB 186-5(Draft). 2019 October. https://doi.org/10.6028/NIST.FIPS.186-5-draft
22. Okeya K., Takagi T., Vuillaume C.: Efficient representations on Koblitz curves with resistance to side channel attacks. ACISP 2005, LNCS, vol. 3574, pp. 218-229. Springer, Heidelberg, 2005.
23. Oliveira T., Aranha D.F., López J., Rodríguez-Henríquez F.: Fast Point Multiplication Algorithms for Binary Elliptic Curves with and without Precomputation. SAC 2014, LNCS, vol. 8781, pp. 324-344. Springer, Heidelberg, 2014.
24. Oliveira T., López J., Aranha D.F., 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.
25. Renes J., Costello C., Batina L.: Complete addition formulas for prime order elliptic curves. EUROCRYPT 2016, LNCS, vol. 9665, pp. 403-428. Springer, Berlin, Heidelberg 2016.

26. Scott M.: MIRACL-Multiprecision integer and rational arithmetic cryptographic library, C/C++ Library, https://github.com/miracl/MIRACL
27. Solinas J.: Efficient arithmetic on Koblitz curves, Des., Codes Cryptography, vol. 19, pp. 195-249, 2000.
28. Trost W. and Xu G.: On the optimal pre-computation of window τNAF for Koblitz curves. IEEE Transactions on Computers. Vol. 65, No. 9. pp. 2918-2924, September 2016.
29. Vuillaume C., Okeya K., Takagi T.: Defeating simple power analysis on Koblitz curves. IEICE Transactions on Fundamentals of Electronics Communications and Computer Sciences, E89-A(5), pp. 1362-1369, 2006.
30. Yu W., Al Musa S., Li B.: Double-base chain for scalar multiplication. EUROCRYPT 2020, Part III, LNCS, vol. 12107, pp. 538-565. Springer, Heidelberg, 2020.

A Pre-Computation for Window τNAF with Widths 7 And 8

A.1 Pre-Computation for Window Width $w = 7$

Our pre-computation on a μ_4-Koblitz curve for window τNAF with width 7 is shown in Table 14. The cost of this pre-computation is 88**M**+62**S** with $a = 0$ and 93**M**+62**S** with $a = 1$.

A.2 Pre-Computation for Window Width $w = 8$

Our pre-computation on a μ_4-Koblitz curve for window τNAF with width 8 is shown in Table 15. The cost of this pre-computation is 186**M**+123**S** with $a = 0$ and 198**M**+123**S** with $a = 1$.

B Our Pre-Computation Scheme on Koblitz Curves Using LD Coordinates

A projective point $P = (X : Y : Z)$ in LD coordinates on an elliptic curve $E/\mathbb{F}_{2^m}$ can be converted to an affine point $(\frac{X}{Z}, \frac{Y}{Z^2})$ [20]. Let $P = (x_P, y_P)$. The projective LD coordinates of P are (X_P, Y_P, Z_P) where $x_P = \frac{X_P}{Z_P}$ and $y_P = \frac{Y_P}{Z_P^2}$. We have $-(x_P, y_P) = (x_P, x_P + y_P)$, $-(X_P, Y_P, Z_P) = (X_P, X_P Z_P + Y_P, Z_P)$, $\tau(x_P, y_P) = (x_P^2, y_P^2)$, and $\tau(X_P, Y_P, Z_P) = (X_P^2, Y_P^2, Z_P^2)$. Let $P = (X_P, Y_P, Z_P)$ and $Q = (X_Q, Y_Q, Z_Q)$. Point addition $P + Q = (x_{P+Q}, \lambda_{P+Q})$ with $Z_P = 1$ was given in Section 3 of [17] as

$$\begin{aligned} A = Z_Q^2 Y_P + Y_Q, B &= Z_Q X_P + X_Q, C = Z_Q B, \\ Z_{P+Q} = C^2, D &= Z_{P+Q} X_P, E = X_P + Y_P, \\ X_{P+Q} &= A^2 + C(A + B^2 + aC), \\ Y_{P+Q} &= (D + X_{P+Q})(AC + Z_{P+Q}) + Z_{P+Q}^2 E. \end{aligned}$$

Table 14. Novel pre-computation for $w = 7$

c_i		Q_i	$a = 0/a = 1$
			88**M**+62**S**/93**M**+62**S**
$c_{37} = -1 + \mu\tau$	$c_{37} = -\mu\bar{\tau}$	$Q_{37} = -\mu\bar{\tau}P$	2**M**+2**S**
$c_{39} = 1 + \mu\tau$	$c_{39} = \mu\bar{\tau}c_{37}$	$Q_{39} = -(\mu\bar{\tau})^2P$	2**M**+2**S**
$c_{35} = -3 + \mu\tau$	$c_{35} = -\mu\bar{\tau}c_{39}$	$Q_{35} = (\mu\bar{\tau})^3P$	2**M**+2**S**
$c_{15} = 1 - 3\mu\tau$	$c_{15} = -\mu\bar{\tau}c_{35}$	$Q_{15} = -(\mu\bar{\tau})^4P$	2**M**+2**S**
$c_{43} = 5 + \mu\tau$	$c_{43} = -\mu\bar{\tau}c_{15}$	$Q_{43} = (\mu\bar{\tau})^5P$	2**M**+2**S**
$c_{53} = 1 - 2\mu\tau$	$c_{53} = \mu\tau + c_{15}$	$Q_{53} = \mu\tau P + Q_{15}$	
$c_{23} = -1 + 4\mu\tau$	$c_{23} = \mu\tau - c_{15}$	$Q_{23} = \mu\tau P - Q_{15}$	(10**M**+3**S**) + 3**S**/(11**M**+3**S**) + 3**S**
$c_{41} = 3 + \mu\tau$	$c_{41} = -\mu\bar{\tau}c_{53}$	$Q_{41} = -\mu\bar{\tau}Q_{53}$	2**M**+2**S**
$c_{19} = 5 - 3\mu\tau$	$c_{19} = \mu\bar{\tau}c_{41}$	$Q_{19} = -(\mu\bar{\tau})^2Q_{53}$	2**M**+2**S**
$c_{63} = 1 + 5\mu\tau$	$c_{63} = -\mu\bar{\tau}c_{19}$	$Q_{63} = (\mu\bar{\tau})^3Q_{53}$	2**M**+2**S**
$c_{27} = -11 + \mu\tau$	$c_{27} = -\mu\bar{\tau}c_{63}$	$Q_{27} = -(\mu\bar{\tau})^4Q_{53}$	2**M**+2**S**
$c_{45} = 7 + \mu\tau$	$c_{45} = \mu\bar{\tau}c_{23}$	$Q_{45} = \mu\bar{\tau}Q_{23}$	2**M**+2**S**
$c_3 = 3$	$c_3 = \mu\tau - c_{35}$	$Q_3 = \mu\tau P - Q_{35}$	
$c_{55} = 3 - 2\mu\tau$	$c_{55} = -\mu\tau - c_{35}$	$Q_{55} = -\mu\tau P - Q_{35}$	10**M**+3**S**/11**M**+3**S**
$c_{17} = 3 - 3\mu\tau$	$c_{17} = \mu\bar{\tau}c_3$	$Q_{17} = \mu\bar{\tau}Q_3$	2**M**+2**S**
$c_{11} = -3 - 3\mu\tau$	$c_{11} = \mu\bar{\tau}c_{17}$	$Q_{11} = (\mu\bar{\tau})^2Q_3$	2**M**+2**S**
$c_{13} = -1 - 3\mu\tau$	$c_{13} = \mu\bar{\tau}c_{55}$	$Q_{13} = \mu\bar{\tau}Q_{55}$	2**M**+2**S**
$c_{31} = -7 + \mu\tau$	$c_{31} = \mu\bar{\tau}c_{13}$	$Q_{31} = (\mu\bar{\tau})^2Q_{55}$	2**M**+2**S**
$c_5 = 5 + 7\mu\tau$	$c_5 = \mu\bar{\tau}c_{31}$	$Q_5 = (\mu\bar{\tau})^3Q_{55}$	2**M**+2**S**
$c_{51} = -1 - 2\mu\tau$	$c_{51} = \mu\tau + c_{13}$	$Q_{51} = \mu\tau P + Q_{13}$	
$c_{25} = 1 + 4\mu\tau$	$c_{25} = \mu\tau - c_{13}$	$Q_{25} = \mu\tau P - Q_{13}$	10**M**+3**S**/11**M**+3**S**
$c_{33} = -5 + \mu\tau$	$c_{33} = \mu\bar{\tau}c_{51}$	$Q_{33} = \mu\bar{\tau}Q_{51}$	2**M**+2**S**
$c_{59} = -3 + 5\mu\tau$	$c_{59} = \mu\bar{\tau}c_{33}$	$Q_{59} = (\mu\bar{\tau})^2Q_{51}$	2**M**+2**S**
$c_7 = -7 - 3\mu\tau$	$c_7 = -\mu\bar{\tau}c_{59}$	$Q_7 = -(\mu\bar{\tau})^3Q_{51}$	2**M**+2**S**
$c_{29} = -9 + \mu\tau$	$c_{29} = -\mu\bar{\tau}c_{25}$	$Q_{29} = -\mu\bar{\tau}Q_{25}$	2**M**+2**S**
$c_{49} = -3 - 2\mu\tau$	$c_{49} = -\mu\tau - c_{41}$	$Q_{49} = -\mu\tau P - Q_{41}$	6**M**+2**S**/7**M**+2**S**
$c_{21} = 7 - 3\mu\tau$	$c_{21} = -\mu\bar{\tau}c_{49}$	$Q_{21} = -\mu\bar{\tau}Q_{49}$	2**M**+2**S**
$c_9 = -1 + 7\mu\tau$	$c_9 = -\mu\bar{\tau}c_{21}$	$Q_9 = (\mu\bar{\tau})^2Q_{49}$	2**M**+2**S**
$c_{57} = 5 - 2\mu\tau$	$c_{57} = -\mu\tau - c_{33}$	$Q_{57} = -\mu\tau P - Q_{33}$	6**M**+2**S**/7**M**+2**S**
$c_{61} = -1 + 5\mu\tau$	$c_{61} = -\mu\bar{\tau}c_{57}$	$Q_{61} = -\mu\bar{\tau}Q_{57}$	2**M**+2**S**
$c_{47} = 9 + \mu\tau$	$c_{47} = \mu\bar{\tau}c_{61}$	$Q_{47} = -(\mu\bar{\tau})^2Q_{57}$	2**M**+2**S**

One full point addition costs 13**M**+4**S**, one mixed point addition costs 8**M**+5**S**, and one point addition with both affine points costs 5**M**+5**S**. Furthermore, evaluation of $-P$ costs 1**M**, evaluation of $\tau(P)$ costs 3**S**, and τ-affine operation requires 2**S**.

B.1 New Formulas Using LD Coordinates

New formulas for $P \pm Q$ We introduce efficient formulas of $P \pm Q$ in LD coordinates by Theorem 5.

Theorem 3 *Let $P = (x_P, y_P)$ and $Q = (X_Q, Y_Q, Z_Q)$ where $P \neq \pm Q$. Notice that $-Q = (X_Q, X_QZ_Q+Y_Q, Z_Q)$. The two operations of $P+Q$ and $P-Q$ (($P\pm Q$)-operation) can be computed as Equation* (5) *at the total cost of* 12**M**+6**S**.

$$
\begin{aligned}
A =& Z_Q^2 Y_P + Y_Q, B = Z_Q X_P + X_Q, C = Z_Q B && \mathbf{3M{+}S} \\
Z_{P+Q} =& Z_{P-Q} = C^2 && \mathbf{S} \\
D =& Z_{P+Q} X_P, E = X_P + Y_P, F = AC && \mathbf{2M} \\
X_{P+Q} =& A^2 + C(A + B^2 + aC) && \mathbf{M{+}2S} \\
Y_{P+Q} =& (D + X_{P+Q})(F + Z_{P+Q}) + Z_{P+Q}^2 E && \mathbf{2M{+}S} \\
G =& X_Q Z_Q C, H = (X_Q Z_Q)^2 + G && \mathbf{2M{+}S} \\
X_{P-Q} =& X_{P+Q} + H && \\
Y_{P-Q} =& Y_{P+Q} + H(G + F + Z_{P+Q}) + (D + X_{P+Q})G && \mathbf{2M}
\end{aligned}
\tag{5}
$$

Theorem 4 ([8]) *Let $P = (X_P, Y_P, Z_P)$ in LD coordinates. $\mu\bar{\tau}P$ can be computed as*

$$
\begin{aligned}
X_{\mu\bar{\tau}P} =& (X_P + Z_P)^2 \\
Z_{\mu\bar{\tau}P} =& X_P Z_P \\
Y_{\mu\bar{\tau}P} =& (Y_P + (1-a)X_{\mu\bar{\tau}P})(Y_P + aX_{\mu\bar{\tau}P} + Z_{\mu\bar{\tau}P}) + (1-a)Z_{\mu\bar{\tau}P}^2
\end{aligned}
$$

at the cost of 2**M**+2**S** *with $a = 0$ and* 2**M**+**S** *with $a = 1$ when $Z_P \neq 1$ and at the cost of* **M**+2**S** *with $a = 0$ and* **M**+**S** *with $a = 1$ when $Z_P = 1$. The cost of $-\mu\bar{\tau}P$ is the same as that of $\mu\bar{\tau}P$.*

B.2 Pre-Computation Schemes Using LD Coordinates

Our pre-computation scheme for window τNAF in LD coordinates is the same as that on a μ_4-Koblitz curve. Our pre-computation scheme requires 5**M**+6**S**, 19**M**+19**S**, 51**M**+40**S**, 99**M**+76**S**, and 214**M**+158**S** when $a = 0$, and 5**M**+3**S**, 19**M**+13**S**, 51**M**+29**S**, 99**M**+53**S**, and 214**M**+113**S** when $a = 1$ using LD coordinates for window τNAF with widths from 4 to 8 respectively.

The costs of different pre-computation schemes for window τNAF with widths from 4 to 6 are summarized in Table 16. Trost and Xu's pre-computation scheme requires 15**M**+19**S**, 48**M**+39**S**, and 120**M**+79**S** for $w = 4$, 5, and 6. Both theoretical analysis and experimental results show that our pre-computation scheme is about 2.4 times faster than Trost and Xu's scheme using LD coordinates.

B.3 Scalar Multiplications Using Window τNAF in LD Coordinates

The Montgomery trick transferring n pre-computations in LD coordinates to affine coordinates costs **I**+$(5n-3)$**M**+n**S**. Let the costs of pre-computation schemes for window τNAF with width w be denoted by PreLD_w.

Constant-time scalar multiplication using window τNAF has two situations.

1. Scalar multiplication uses pre-computations in LD coordinates. It requires m τ-operations, $\frac{m}{w-1}$ point additions, the pre-computation, and negative of the pre-computation. Scalar multiplication is expected to cost

$$3m\mathbf{S} + \frac{m}{w-1}(13\mathbf{M} + 4\mathbf{S}) + \mathrm{PreLD}_w + (2^{w-2} - 1)\mathbf{M}.$$

2. Scalar multiplication uses pre-computations in affine coordinates. It requires m τ-projective operations, $\frac{m}{w-1}$ mixed additions, Montgomery trick, and the pre-computation. Scalar multiplication is expected to cost

$$3m\mathbf{S} + \frac{m}{w-1}(8\mathbf{M} + 5\mathbf{S}) + \mathbf{I} + (5 \cdot 2^{w-2} - 8)\mathbf{M} + (2^{w-2} - 1)\mathbf{S} + \mathrm{PreLD}_w.$$

We summarize the lowest costs of constant-time scalar multiplications on K1-163, K-233, K-283, K1-283, K-409, and K-571 using our pre-computation scheme in Table 17. Our experimental results show that our constant-time scalar multiplication on Koblitz curves using LD coordinates saves up to 10% compared to Trost and Xu's work using LD coordinates.

C Our Pre-Computation Scheme on Koblitz Curves Using $\boldsymbol{\lambda}$-Coordinates

Given an affine point $P = (x, y)$ on an elliptic curve $E/\mathbb{F}_{2^m}$, its lambda representation is (x, λ) with $\lambda = x + \frac{y}{x}$ [24]. Let $P = (x_P, \lambda_P)$ with $\lambda_P = x_P + \frac{y_P}{x_P}$. The λ-coordinates of $-P$ are $(x_P, \lambda_P + 1)$. The λ-projective coordinates of P are (X_P, Λ_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 Section 3.1 of [24] as

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

One full point addition costs 11**M**+2**S**, one mixed point addition costs 8**M**+2**S** and one point addition with both affine points costs 5**M**+2**S**. Furthermore, evaluation of $\tau(P)$ costs 3**S** and τ-affine operation requires 2**S**.

C.1 New Formulas Using $\boldsymbol{\lambda}$-Coordinates

New formulas for $\boldsymbol{P \pm Q}$ We introduce efficient formulas of $P \pm Q$ in λ-projective coordinates by Theorem 5.

Theorem 5 *Let $P = (x_P, \lambda_P)$ and $Q = (X_Q, \Lambda_Q, Z_Q)$ where $P \neq \pm Q$. Notice that $-Q = (X_Q, \Lambda_Q + Z_Q, Z_Q)$. The two operations of $P+Q$ and $P-Q$ (($P \pm Q$)-operation) can be computed as Equation* (6) *at the total cost of* 12**M**+5**S**.

$$
\begin{aligned}
A =& \lambda_P Z_Q + \Lambda_Q && \mathbf{M} \\
B =& (x_P Z_Q + X_Q)^2 && \mathbf{M}\text{+}\mathbf{S} \\
C =& X_Q Z_Q && \mathbf{M} \\
D =& x_P C && \mathbf{M} \\
X_{P+Q} =& A^2 D && \mathbf{M}\text{+}\mathbf{S} \\
Z_{P+Q} =& BAZ_Q && 2\mathbf{M} \\
\Lambda_{P+Q} =& (AX_Q + B)^2 + Z_{P+Q}(\lambda_P + 1) && 2\mathbf{M}\text{+}\mathbf{S} \\
X_{P-Q} =& X_{P+Q} + DZ_Q^2 && \mathbf{M}\text{+}\mathbf{S} \\
Z_{P-Q} =& Z_{P+Q} + BZ_Q^2 && \mathbf{M} \\
\Lambda_{P-Q} =& \Lambda_{P+Q} + C^2 + BZ_Q^2(\lambda_P + 1) && \mathbf{M}\text{+}\mathbf{S}
\end{aligned} \tag{6}
$$

Formulas for $\mu\bar{\tau}$-operations An efficient formula for $\mu\bar{\tau}P$ in λ-coordinates has been obtained in Section 4 of [28] under the form of $P - \mu\tau P$. We shall use their formula $\mu\bar{\tau}P = \left(\frac{x_P^2+1}{x_P}, \frac{x_P^2}{x_P^2+1} + \lambda_P\right)$ with $P = (x_P, \lambda_P)$. Formulas for $(\mu\bar{\tau})^2 P$ and $(\mu\bar{\tau})^3 P$ were also reported in Section 4 of [28] under the form of $P + \mu\tau P$ and $P - \tau^2 P$, however in [28], these formulas were not based on the one for $\mu\bar{\tau}P$. We can get a good improvement by designing efficient formulas of $(\mu\bar{\tau})^i P$ by utilizing $(\mu\bar{\tau})^{i-1} P$ if it is already computed.

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

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 Section 4.1 of [28], 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 $\mu\bar{\tau}P$ can be calculated as Equation (7) at the cost of 5**M**+3**S**.

$$
\begin{aligned}
\alpha =& X_P Z_P && \mathbf{M} \\
A_1 =& X_P^2 + Z_P^2 && 2\mathbf{S} \\
X_{\mu\bar{\tau}P} =& A_1^2 && \mathbf{S} \\
\Lambda_{\mu\bar{\tau}P} =& \alpha X_P^2 + X_P \Lambda_P A_1 && 3\mathbf{M} \\
Z_{\mu\bar{\tau}P} =& A_1 \alpha && \mathbf{M}
\end{aligned} \tag{7}
$$

2. The values for computing previous point operations are utilized to compute a new point operation in [19,28]. Motivated by their trick, 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 Equation (7) where $x_{\mu\bar{\tau}P} = \frac{A_1}{\alpha}$. Notice that $(\mu\bar{\tau})^2 P =$

$\mu\bar{\tau}(\mu\bar{\tau}P)$. We have

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

Then $(\mu\bar{\tau})^2P$ can be computed as Equation (8) at the cost of 3**M**+2**S**.

$$\begin{aligned}A_2 &= X_{\mu\bar{\tau}P}+\alpha^2 && \mathbf{S}\\ X_{(\mu\bar{\tau})^2P} &= A_2^2 && \mathbf{S}\\ \Lambda_{(\mu\bar{\tau})^2P} &= X_{\mu\bar{\tau}P}Z_{\mu\bar{\tau}P}+\Lambda_{\mu\bar{\tau}P}A_2 && 2\mathbf{M}\\ Z_{(\mu\bar{\tau})^2P} &= Z_{\mu\bar{\tau}P}A_2 && \mathbf{M}\end{aligned} \tag{8}$$

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

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

Some values of calculating $(\mu\bar{\tau})^{i-1}P$ are used to calculate $(\mu\bar{\tau})^iP$. When $i = 3$, $x_{(\mu\bar{\tau})^{i-1}P} = \frac{A_{i-1}}{Z_{(\mu\bar{\tau})^{i-2}P}}$ and $X_{(\mu\bar{\tau})^{i-1}P} = A_{i-1}^2$ are computed by Equation (8); when $i > 3$, $x_{(\mu\bar{\tau})^{i-1}P} = \frac{A_{i-1}}{Z_{(\mu\bar{\tau})^{i-2}P}}$ and $X_{(\mu\bar{\tau})^{i-1}P} = A_{i-1}^2$ are computed by Equation (9). $(\mu\bar{\tau})^iP$ can be computed as

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

Then $(\mu\bar{\tau})^iP$, $i \geq 3$ can be computed as Equation (9) at the cost of 3**M**+2**S**.

$$\begin{aligned}A_i &= X_{(\mu\bar{\tau})^{i-1}P}+Z_{(\mu\bar{\tau})^{i-2}P}^2 && \mathbf{S}\\ X_{(\mu\bar{\tau})^iP} &= A_i^2 && \mathbf{S}\\ \Lambda_{(\mu\bar{\tau})^iP} &= X_{(\mu\bar{\tau})^{i-1}P}Z_{(\mu\bar{\tau})^{i-1}P}+\Lambda_{(\mu\bar{\tau})^{i-1}P}A_i && 2\mathbf{M}\\ Z_{(\mu\bar{\tau})^iP} &= Z_{(\mu\bar{\tau})^{i-1}P}A_i && \mathbf{M}\end{aligned} \tag{9}$$

□

Notice that $\mu\bar{\tau}P = P - \mu\tau P$, $(\mu\bar{\tau})^2P = -(P + \mu\tau P)$, and $(\mu\bar{\tau})^3P = -(P - \tau^2P)$. Trost and Xu showed that $P - \mu\tau P$, $P + \mu\tau P$, and $P - \tau^2P$ cost 5**M**+3**S**, 7**M**+5**S**, and 5**M**+3**S** respectively. Their formula of $P-\mu\tau P$ is still the state-of-the-art. The costs of $(\mu\bar{\tau})^2P$ and $(\mu\bar{\tau})^3P$ are 3**M**+2**S** and 3**M**+2**S** which largely improves their costs of 7**M**+5**S** and 5**M**+3**S**.

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 Equation (8), the formulas of $\mu\bar{\tau}P$ and $(\mu\bar{\tau})^2P$ are shown as Equation (10) at the total cost of 4**M**+3**S**.

$$\begin{aligned}
\beta &= x_P^2 && \mathbf{S}\\
X_{\mu\bar{\tau}P} &= \beta^2 + 1 && \mathbf{S}\\
Z_{\mu\bar{\tau}P} &= x_P\beta + x_P && \mathbf{M}\\
\Lambda_{\mu\bar{\tau}P} &= (\lambda_P + 1)Z_{\mu\bar{\tau}P} + x_P && \mathbf{M}\\
A_2 &= X_{\mu\bar{\tau}P} + \beta && \\
X_{(\mu\bar{\tau})^2P} &= A_2^2 && \mathbf{S}\\
Z_{(\mu\bar{\tau})^2P} &= A_2Z_{\mu\bar{\tau}P} && \mathbf{M}\\
\Lambda_{(\mu\bar{\tau})^2P} &= Z_{(\mu\bar{\tau})^2P}\lambda_P + x_P && \mathbf{M}
\end{aligned} \tag{10}$$

C.2 Pre-Computation Schemes Using λ-Coordinates

Our pre-computation scheme for window τNAF in λ-coordinates is the same as that on a μ_4-Koblitz curve except Q_5 and Q_7 for window width 6 in Table 7 and Q_{35} for window width 8 in Table 15. Q_5 and Q_7 are computed as $Q_5 = \mu\tau P + Q_{31}$ and $Q_7 = \mu\tau P - Q_{31}$ with $c_5 = -7 + 2\mu\tau$ and $c_7 = 7$ by one $(P \pm Q)$-operation. Q_{35} is computed as $\mu\tau P + Q_{125}$ with $c_{35} = -13 + 8\mu\tau$. Our pre-computation scheme requires 7**M**+5**S**, 26**M**+16**S**, 66**M**+36**S**, 135**M**+72**S**, and 282**M**+148**S** using λ-projective coordinates for window τNAF with widths from 4 to 8 respectively.

The costs of different pre-computation schemes for window τNAF with widths from 4 to 6 are summarized in Table 18. Trost and Xu's pre-computation scheme requires 12**M**+8**S**, 44**M**+18**S**, and 108**M**+36**S** for $w = 4$, 5, and 6 based on their efficient formulas for $P - \mu\tau(P)$, $P + \mu\tau(P)$ and $P - \tau^2(P)$. Both theoretical analysis and experimental results show that our pre-computation scheme is about 40% faster than Trost and Xu's scheme using λ-coordinates.

C.3 Scalar Multiplications Using Window τNAF in λ-Coordinates

The Montgomery trick transferring n pre-computations in λ-projective coordinates to λ-coordinates costs **I**+$(5n - 3)$**M**. Let the costs of pre-computation schemes for window τNAF with width w be denoted by $\mathrm{Pre}\lambda_w$.

Constant-time scalar multiplication using window τNAF has two situations.

1. Scalar multiplication uses pre-computations in λ-projective coordinates. It requires m τ-operations, $\frac{m}{w-1}$ point additions, and the pre-computation. Scalar multiplication is expected to cost

$$3m\mathbf{S} + \frac{m}{w-1}(11\mathbf{M} + 2\mathbf{S}) + \mathrm{Pre}\lambda_w.$$

2. Scalar multiplication uses pre-computations in λ-coordinates. It requires m τ-projective operations, $\frac{m}{w-1}$ mixed additions, Montgomery trick, and the pre-computation. Scalar multiplication is expected to cost

$$3m\mathbf{S} + \frac{m}{w-1}(8\mathbf{M} + 2\mathbf{S}) + \mathbf{I} + (5 \cdot 2^{w-2} - 8)\mathbf{M} + \mathrm{Pre}\lambda_w.$$

We summarize the lowest costs of constant-time scalar multiplications on K1-163, K-233, K-283, K1-283, K-409, and K-571 using our pre-computation scheme in Table 19. Our experimental results show that our constant-time scalar multiplication on Koblitz curves using λ-coordinates saves up to 6.5% compared to Trost and Xu's work using λ-coordinates.

Table 15. Novel pre-computation for $w = 8$

c_i		Q_i	$a = 0/a = 1$
			186M+123S/198M+123S
$c_{91} = 1 - \mu\tau$	$c_{91} = \mu\bar{\tau}$	$Q_{91} = \mu\bar{\tau}P$	2M+2S
$c_{89} = -1 - \mu\tau$	$c_{89} = \mu\bar{\tau}c_{91}$	$Q_{89} = (\mu\bar{\tau})^2P$	2M+2S
$c_{93} = 3 - \mu\tau$	$c_{93} = -\mu\bar{\tau}c_{89}$	$Q_{93} = -(\mu\bar{\tau})^3P$	2M+2S
$c_{15} = 1 - 3\mu\tau$	$c_{15} = \mu\bar{\tau}c_{93}$	$Q_{15} = -(\mu\bar{\tau})^4P$	2M+2S
$c_{85} = -5 - \mu\tau$	$c_{85} = \mu\bar{\tau}c_{15}$	$Q_{85} = -(\mu\bar{\tau})^5P$	2M+2S
$c_{55} = -7 + 5\mu\tau$	$c_{55} = \mu\bar{\tau}c_{85}$	$Q_{55} = -(\mu\bar{\tau})^6P$	2M+2S
$c_{115} = -3 - 7\mu\tau$	$c_{115} = -\mu\bar{\tau}c_{55}$	$Q_{115} = (\mu\bar{\tau})^7P$	2M+2S
$c_{75} = -1 + 2\mu\tau$	$c_{75} = -\mu\tau - c_{15}$	$Q_{75} = -\mu\tau P - Q_{15}$	
$c_{105} = 1 - 4\mu\tau$	$c_{105} = -\mu\tau + c_{15}$	$Q_{105} = -\mu\tau P + Q_{15}$	10M+3S+3S/11M+3S+3S
$c_{87} = -3 - \mu\tau$	$c_{87} = -\mu\bar{\tau}c_{75}$	$Q_{87} = -\mu\bar{\tau}Q_{75}$	2M+2S
$c_{19} = 5 - 3\mu\tau$	$c_{19} = -\mu\bar{\tau}c_{87}$	$Q_{19} = (\mu\bar{\tau})^2Q_{75}$	2M+2S
$c_{63} = 1 + 5\mu\tau$	$c_{63} = -\mu\bar{\tau}c_{19}$	$Q_{63} = -(\mu\bar{\tau})^3Q_{75}$	2M+2S
$c_{101} = 11 - \mu\tau$	$c_{101} = \mu\bar{\tau}c_{63}$	$Q_{101} = -(\mu\bar{\tau})^4Q_{75}$	2M+2S
$c_{25} = -9 + 11\mu\tau$	$c_{25} = -\mu\bar{\tau}c_{101}$	$Q_{25} = (\mu\bar{\tau})^5Q_{75}$	2M+2S
$c_{83} = -7 - \mu\tau$	$c_{83} = \mu\bar{\tau}c_{105}$	$Q_{83} = \mu\bar{\tau}Q_{105}$	2M+2S
$c_{127} = 9 - 7\mu\tau$	$c_{127} = -\mu\bar{\tau}c_{83}$	$Q_{127} = -(\mu\bar{\tau})^2Q_{105}$	2M+2S
$c_{37} = -5 - 9\mu\tau$	$c_{37} = \mu\bar{\tau}c_{127}$	$Q_{37} = -(\mu\bar{\tau})^3Q_{105}$	2M+2S
$c_3 = 3$	$c_3 = -\mu\tau - c_{87}$	$Q_3 = -\mu\tau P - Q_{87}$	
$c_{79} = 3 + 2\mu\tau$	$c_{79} = \mu\tau - c_{87}$	$Q_{79} = \mu\tau P - Q_{87}$	10M+3S/11M+3S
$c_{17} = 3 - 3\mu\tau$	$c_{17} = \mu\bar{\tau}c_3$	$Q_{17} = \mu\bar{\tau}Q_3$	2M+2S
$c_{11} = -3 - 3\mu\tau$	$c_{11} = \mu\bar{\tau}c_{17}$	$Q_{11} = (\mu\bar{\tau})^2Q_3$	2M+2S
$c_{23} = 9 - 3\mu\tau$	$c_{23} = -\mu\bar{\tau}c_{11}$	$Q_{23} = -(\mu\bar{\tau})^3Q_3$	2M+2S
$c_{45} = 3 - 9\mu\tau$	$c_{45} = \mu\bar{\tau}c_{23}$	$Q_{45} = -(\mu\bar{\tau})^4Q_3$	2M+2S
$c_{21} = 7 - 3\mu\tau$	$c_{21} = \mu\bar{\tau}c_{49}$	$Q_{21} = \mu\bar{\tau}Q_{79}$	2M+2S
$c_{119} = 1 - 7\mu\tau$	$c_{119} = \mu\bar{\tau}c_{21}$	$Q_{119} = (\mu\bar{\tau})^2Q_{79}$	2M+2S
$c_{73} = -3 + 2\mu\tau$	$c_{73} = -\mu\tau - c_{17}$	$Q_{73} = -\mu\tau P - Q_{17}$	
$c_{107} = 3 - 4\mu\tau$	$c_{107} = -\mu\tau + c_{17}$	$Q_{107} = -\mu\tau P + Q_{17}$	10M+3S/11M+3S
$c_{13} = -1 - 3\mu\tau$	$c_{13} = -\mu\bar{\tau}c_{73}$	$Q_{13} = -\mu\bar{\tau}Q_{73}$	2M+2S
$c_{97} = 7 - \mu\tau$	$c_{97} = -\mu\bar{\tau}c_{13}$	$Q_{97} = (\mu\bar{\tau})^2Q_{73}$	2M+2S
$c_{123} = 5 - 7\mu\tau$	$c_{123} = \mu\bar{\tau}c_{97}$	$Q_{123} = (\mu\bar{\tau})^3Q_{73}$	2M+2S
$c_9 = -5 - 3\mu\tau$	$c_9 = \mu\bar{\tau}c_{107}$	$Q_9 = \mu\bar{\tau}Q_{107}$	2M+2S
$c_{51} = -11 + 5\mu\tau$	$c_{51} = \mu\bar{\tau}c_9$	$Q_{51} = (\mu\bar{\tau})^2Q_{107}$	2M+2S
$c_{33} = -1 + 11\mu\tau$	$c_{33} = \mu\bar{\tau}c_{51}$	$Q_{33} = (\mu\bar{\tau})^3Q_{107}$	2M+2S
$c_{77} = 1 + 2\mu\tau$	$c_{77} = -\mu\tau - c_{13}$	$Q_{77} = -\mu\tau P - Q_{13}$	
$c_{103} = -1 - 4\mu\tau$	$c_{103} = -\mu\tau + c_{13}$	$Q_{103} = -\mu\tau P + Q_{13}$	10M+3S/11M+3S
$c_{95} = 5 - \mu\tau$	$c_{95} = \mu\bar{\tau}c_{77}$	$Q_{95} = \mu\bar{\tau}Q_{77}$	2M+2S
$c_{59} = -3 + 5\mu\tau$	$c_{59} = -\mu\bar{\tau}c_{95}$	$Q_{59} = -(\mu\bar{\tau})^2Q_{77}$	2M+2S
$c_7 = -7 - 3\mu\tau$	$c_7 = -\mu\bar{\tau}c_{59}$	$Q_7 = (\mu\bar{\tau})^3Q_{77}$	2M+2S
$c_{125} = -13 + 7\mu\tau$	$c_{125} = \mu\bar{\tau}c_7$	$Q_{125} = (\mu\bar{\tau})^4Q_{77}$	2M+2S
$c_{99} = 9 - \mu\tau$	$c_{99} = -\mu\bar{\tau}c_{103}$	$Q_{99} = -\mu\bar{\tau}Q_{103}$	2M+2S
$c_{49} = 7 - 9\mu\tau$	$c_{49} = \mu\bar{\tau}c_{99}$	$Q_{49} = -(\mu\bar{\tau})^2Q_{103}$	2M+2S
$c_5 = 5$	$c_5 = -\mu\tau - c_{85}$	$Q_5 = -\mu\tau P - Q_{85}$	6M+2S/7M+2S
$c_{57} = -5 + 5\mu\tau$	$c_{57} = -\mu\bar{\tau}c_5$	$Q_{57} = -\mu\bar{\tau}Q_5$	2M+2S
$c_{67} = 5 + 5\mu\tau$	$c_{67} = \mu\bar{\tau}c_{57}$	$Q_{67} = -(\mu\bar{\tau})^2Q_5$	2M+2S
$c_{47} = -15 + 5\mu\tau$	$c_{47} = -\mu\bar{\tau}c_{67}$	$Q_{47} = (\mu\bar{\tau})^3Q_5$	2M+2S
$c_{71} = -5 + 2\mu\tau$	$c_{71} = -\mu\tau - c_{19}$	$Q_{71} = -\mu\tau P - Q_{19}$	
$c_{61} = -1 + 5\mu\tau$	$c_{61} = \mu\bar{\tau}c_{71}$	$Q_{61} = \mu\bar{\tau}Q_{71}$	2M+2S
$c_{109} = 5 - 4\mu\tau$	$c_{109} = -\mu\tau + c_{19}$	$Q_{109} = -\mu\tau P + Q_{19}$	10M+3S/11M+3S
$c_{81} = -9 - \mu\tau$	$c_{81} = -\mu\bar{\tau}c_{61}$	$Q_{81} = -(\mu\bar{\tau})^2Q_{71}$	2M+2S
$c_{53} = 11 - 9\mu\tau$	$c_{53} = -\mu\bar{\tau}c_{81}$	$Q_{53} = (\mu\bar{\tau})^3Q_{71}$	2M+2S
$c_{65} = 3 + 5\mu\tau$	$c_{65} = -\mu\bar{\tau}c_{109}$	$Q_{65} = -\mu\bar{\tau}Q_{109}$	2M+2S
$c_{27} = 13 - 3\mu\tau$	$c_{27} = \mu\bar{\tau}c_{65}$	$Q_{27} = -(\mu\bar{\tau})^2Q_{109}$	2M+2S
$c_{69} = -7 + 2\mu\tau$	$c_{69} = -\mu\tau - c_{21}$	$Q_{69} = -\mu\tau P - Q_{21}$	
$c_{111} = 7 - 4\mu\tau$	$c_{111} = -\mu\tau + c_{21}$	$Q_{111} = -\mu\tau P + Q_{21}$	10M+3S/11M+3S
$c_{121} = 3 - 7\mu\tau$	$c_{121} = -\mu\bar{\tau}c_{69}$	$Q_{121} = -\mu\bar{\tau}Q_{69}$	2M+2S
$c_{117} = -1 - 7\mu\tau$	$c_{117} = \mu\bar{\tau}c_{111}$	$Q_{117} = \mu\bar{\tau}Q_{111}$	2M+2S
$c_{113} = 9 - 4\mu\tau$	$c_{113} = -\mu\tau + c_{23}$	$Q_{113} = -\mu\tau P + Q_{23}$	6M+2S/7M+2S
$c_{43} = 1 - 9\mu\tau$	$c_{43} = \mu\bar{\tau}c_{113}$	$Q_{43} = \mu\bar{\tau}Q_{113}$	2M+2S
$c_{39} = 11 - 6\mu\tau$	$c_{39} = -\mu\tau - c_{51}$	$Q_{39} = -\mu\tau P - Q_{51}$	6M+2S/7M+2S
$c_{35} = 1 + 11\mu\tau$	$c_{35} = -\mu\bar{\tau}c_{39}$	$Q_{35} = -\mu\bar{\tau}Q_{39}$	2M+2S
$c_{29} = 1 - 6\mu\tau$	$c_{29} = -\mu\tau - c_{61}$	$Q_{29} = -\mu\tau P - Q_{61}$	6M+2S/7M+2S
$c_{31} = 3 - 6\mu\tau$	$c_{31} = -\mu\tau - c_{59}$	$Q_{31} = -\mu\tau P - Q_{59}$	6M+2S/7M+2S
$c_{41} = 13 - 6\mu\tau$	$c_{41} = \mu\tau - c_{125}$	$Q_{41} = \mu\tau P - Q_{125}$	6M+2S/7M+2S

Table 16. Cost of pre-computations using LD coordinates with $a = 0/a = 1$

	$w = 4$	$w = 5$	$w = 6$
Solinas	15**M**+21**S**	45**M**+52**S**	-
Hankerson, Menezes, Vanstone	15**M**+21**S**	54**M**+49**S**	125**M**+105**S**
Trost, Xu	15**M**+19**S**	48**M**+39**S**	120**M**+79**S**
Ours	5**M**+6**S**/5**M**+3**S**	19**M**+19**S**/19**M**+13**S**	51**M**+40**S**/51**M**+29**S**

Table 17. The expected costs of constant-time scalar multiplications using our pre-computation in LD coordinates on K1-163, K-233, K-283/K1-283, K-409, and K-571 in **M**

		K1-163(w)	K-233(w)	K-283(w)/K1-283(w)	K-409(w)	K-571(w)
S=0**M**	regular τNAF	1304	1864	2264	3272	4568
	Trost, Xu	416(5,M)	556(5,M)	654.8(6,M)	856.4(6,M)	1115.6(6,M)
	Ours	387(5,M)	505.8(6,M)	585.8(6,M)	787.4(6,M)	1022.3(7,M)
S=0.2**M**	regular τNAF	1564.8	2236.8	2716.8	3926.4	5481.6
	Trost, Xu	563.8(5,M)	763.3(5,M)	900(6,M)	1202.4(6,M)	1591.2(6,M)
	Ours	529.6(5,M)	703.2(6,M)	823.2(6,M)/821(6,M)	1125.6(6,M)	1481.5(7,M)
S=0.5**M**	regular τNAF	1956	2796	3396	4908	6852
	Trost, Xu	908(6)	1214.1(5,M)	1407.8(6,M)	1861.4(6,M)	2444.6(6,M)
	Ours	808.5(6)	1100(7)	1300(7)/1288.5(7)	1772.9(6,M)	2310.3(7,M)

Table 18. Cost of pre-computations using λ-coordinates

	$w = 4$	$w = 5$	$w = 6$
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	12**M**+8**S**	44**M**+18**S**	108**M**+36**S**
Ours	7**M**+5**S**	26**M**+16**S**	66**M**+36**S**

Table 19. The expected costs of constant-time scalar multiplications using our pre-computation in λ-coordinates on K1-163, K-233, K-283/K1-283, K-409, and K-571 in **M**

		K1-163(w)	K-233(w)	K-283/K1-283(w)	K-409(w)	K-571(w)
S=0**M**	regular τNAF	1304	1864	2264	3272	4568
	Trost, Xu	412(5,M)	552(5,M)	642.8(6,M)	844.4(6,M)	1103.6(6,M)
	Ours	394(5,M)	520.8(6,M)	600.8(6,M)	802.4(6,M)	1058.3(7,M)
S=0.2**M**	regular τNAF	1467	2097	2547	3681	5139
	Trost, Xu	529.7(5,M)	718.7(5,M)	842.4(6,M)	1129.7(6,M)	1499.1(6,M)
	Ours	511.3(5,M)	686.4(6,M)	800.4 (6,M)	1087.7(6,M)	1453.4(7,M)
S=0.5**M**	regular τNAF	1711.5	2446.5	2971.5	4294.5	5995.5
	Trost, Xu	755.6(6)	1026(6)	1219.1(6)	1697.7(6,M)	2232.3(6,M)
	Ours	713.6(6)	982.9(7)	1157.1(7)	1596.1(7)	2160.6(7)