Speeding Up the Fixed-Base Comb Method for Faster Scalar Multiplication on Koblitz Curves

Christian Hanser and Christian Wagner

Institute for Applied Information Processing and Communications (IAIK),
Graz University of Technology (TUG), Inffeldgasse 16a, 8010 Graz, Austria
{christian.hanser|christian.wagner@iaik.tugraz.at}

Abstract. Scalar multiplication is the most expensive arithmetical operation on elliptic curves. There are various methods available, which are optimized for different settings, such as high speed, side-channel resistance and small memory footprint. One of the fastest methods for fixed-base scalar multiplications is the so-called fixed-base comb scalar multiplication method, which is due to Lim and Lee. In this paper, we present a modification to this method, which exploits the possibility of exchanging doublings for much cheaper applications of the Frobenius endomorphism on binary Koblitz curves. We have implemented the findings in software and compare the performance of the implementation to the performance of the reference WTNAF implementation and the performance of the conventional comb multiplication methods. For single scalar multiplications, we are able to achieve performance improvements over the WTNAF method of up to 25% and of up to 42% over the conventional comb methods. Finally, we emphasize that the implementation of the τ-comb method is straight-forward and requires only little effort. All in all, this makes it a good alternative to other fixed-base multiplication methods.

Keywords: ECC, scalar multiplication, Lim-Lee method, comb method, Koblitz curves, Frobenius endomorphism, τ-adic representation

1 Introduction

In 1985, Neal Koblitz [8] and Victor S. Miller [14] both discovered independently that elliptic curves over finite fields can be used for public key cryptography. By now, elliptic curves have become an important concept within public key cryptography. This is mainly due to small key sizes compared to other public key cryptosystems, such as RSA, which lower the requirements for CPUs as well as the memory footprint. Elliptic curves have a group structure and their cryptographic security relies on the difficulty of solving the elliptic curve discrete logarithm problem (ECDLP).

The most expensive, yet, at the same time most important operation, on elliptic curves is the scalar multiplication. There are plenty of different techniques achieving optimizations for different settings, such as high speed through the

reduction of group operations, side-channel resistance and small memory footprint. In this paper we are focusing solely on the performance aspects of scalar multiplication. Koblitz curves allow trading the doubling operation for another operation, that is, the application of the so-called Frobenius endomorphism, which is inexpensive compared to point doubling. In the context of Koblitz curves, many multiplication methods relying on a non-adjacent form (NAF), such as windowed non-adjacent form (WNAF), interleaved simultaneous point multiplication, etc., have been adapted, so that doublings can be replaced by far more efficient applications of the Frobenius endomorphism. Yet, to the best of our knowledge, no such modification to the fixed-base comb method, which is also referred to as Lim-Lee method [10], has been proposed until now. This algorithm achieves very high speed at the cost of precomputations and increased memory usage due to lookup tables.

1.1 Contribution

In this paper, we are going to introduce an improvement of the fixed-base comb method on Koblitz curves. In this context, we show how point doublings can be traded for applications of the Frobenius endomorphism. To do so, we use scalar recoding to obtain an unsigned τ-adic representation of scalars, which allows us to exchange the doubling steps within the comb methods for the application of the Frobenius endomorphism. We detail the scalar recoding algorithm and describe the modified fixed-base comb method. Furthermore, we have implemented this method as well as the WTNAF method in software. We give a detailed performance comparison and illustrate the performance gains we have achieved with respect to the WTNAF and the conventional fixed-base comb methods. In the former case, we achieve performance improvements of up to 25% and in the latter case improvements of up to 42%.

1.2 Outline

In Section 2, we are going to talk about elliptic curves in general, binary Koblitz curves and the fixed-base comb multiplication methods. Section 3 discusses the improvements we have achieved and Section 4 details the implementation, presents the benchmark results and compares the findings to the conventional comb multiplier variants. At last, Section 5 concludes the paper.

2 Preliminaries

An elliptic curve over a field $\mathbb{K}$ is a plane, smooth curve described by the Weierstrass equation:

$$E : Y^2 + a'_1XY + a'_3Y = X^3 + a'_2X^2 + a'_4X + a'_6, \tag{1}$$

where $a'_1, a'_2, a'_3, a'_4, a'_6 \in \mathbb{K}$. The set $E(\mathbb{K})$ of points $(x, y) \in \mathbb{K}^2$ satisfying Equation (1) plus the point at infinity $\mathcal{O} = (0 : 1 : 0)$, which is the neutral element,

forms an additive Abelian group. The addition of points on an elliptic curve is achieved using the *chord-and-tangent* method [17].

For cryptographic purposes, one is interested in elliptic curves over finite fields $\mathbb{F}_q$, especially in curves over fields $\mathbb{F}_{2^m}$ with m being prime and $\mathbb{F}_p$ with p being a large prime number. The security of elliptic curve cryptography is based on the assumption that in general there is no subexponential-time algorithm to solve the discrete logarithm problem in elliptic curve groups (ECDLP). Given $P, Q \in E(\mathbb{F}_q)$ with P being a generator of a subgroup of $E(\mathbb{F}_q)$ of large prime order n, the ECDLP constitutes the problem of finding a scalar $k \in \mathbb{Z}_n$ such that $Q = k \cdot P$. In the following, we use $E(\mathbb{F}_q)[n]$ to denote this n-order subgroup of $E(\mathbb{F}_q)$.

In cryptography, we usually use non-supersingular curves, as supersingular curves turned out to be susceptible to the MOV attack [13], which reduces the ECDLP to the DLP on finite fields for which an efficient subexponential-time algorithm is known. Non-supersingular curves defined over $\mathbb{F}_{2^m}$, are described using the following short form of Equation (1):

$$Y^2 + XY = X^3 + a_2X^2 + a_6, \tag{2}$$

where $a_2, a_6 \in \mathbb{F}_{2^m}$ and $a_6 \neq 0$.

The most important and at the same time most expensive operation on elliptic curves is the scalar multiplication, that is, computing $k \cdot P$ from some point $P \in E(\mathbb{F}_q)$ and some scalar $k \in \mathbb{Z}_n$. For this purpose, many different algorithms for different goals, such as small or even constant memory footprint, high speed or side-channel resistance, have been invented.

Basic algorithms for scalar multiplication are the double-and-add and the NAF method. In the former case, one is given an unsigned binary representation of the scalar and per bit of the scalar a doubling is performed, where for bits set to 1 an additional point addition is carried out. In the latter, the unsigned scalar representation is replaced by a signed representation, i.e., bits can now take values in $\{0, \pm 1\}$, where for negative bit values point subtractions instead of additions have to be performed. This way, the Hamming weight of the scalar can be reduced to 1/3, which gives a major speedup by reducing the number of additions. Further speed-ups can be achieved by reducing the number of additions, which is done by the windowed-NAF (WNAF) method at the expense of some precomputations (cf. [6, 4]).

Subsequently, we are dealing with so-called fixed-base comb multipliers, which are very fast methods for multiplications with some fixed-point P (mostly a fixed generator of $E(\mathbb{F}_q)[n]$), and with scalar multiplications on Koblitz curves, for which enormous speed improvements exist already.

2.1 Fixed-Base Comb Multiplication

The fixed-base comb multiplier is due to Lim and Lee [10] and is one of the fastest scalar multiplication methods available. As it requires intense precomputations, it is only used for multiplications with a fixed point, which is in most cases a generator of an elliptic curve (sub-)group.

In the fixed-base comb method, the bit representation of a scalar k of maximum bitsize t (where t is the curve's group order in bits) is prepended with 0s, such that the length of k is a multiple of the window size w and then split into w blocks K_i with $0 \leq i < w$ of size $d = \lceil t/w \rceil$, so that

$$k = K_{w-1} \| \cdots \| K_1 \| K_0.$$

Each bitstring K_i represents one row of a binary matrix with w rows and d columns:

$$\begin{bmatrix} K_0 \\ \vdots \\ K_i \\ \vdots \\ K_{w-1} \end{bmatrix} = \begin{bmatrix} K_{0,d-1} & \cdots & K_{0,0} \\ \vdots & & \vdots \\ K_{i,d-1} & \cdots & K_{i,0} \\ \vdots & & \vdots \\ K_{w-1,d-1} & \cdots & K_{w-1,0} \end{bmatrix} = \begin{bmatrix} k_{d-1} & \cdots & k_0 \\ \vdots & & \vdots \\ k_{(i+1)d-1} & \cdots & k_{id} \\ \vdots & & \vdots \\ k_{wd-1} & \cdots & k_{(w-1)d} \end{bmatrix} \tag{3}$$

Here, $K_{j,i}$ denotes the i-th bit of the bitstring K_j and k_i denotes the i-th bit of the unsigned binary representation of the scalar k. The columns of this matrix are then processed one at a time. By precomputing the points

$$[a_{w-1}, \ldots, a_2, a_1, a_0]_2 P = a_{w-1} 2^{(w-1)d} P + \cdots + a_2 2^{2d} P + a_1 2^d P + a_0 P,$$

for all possible bitstrings $(a_{w-1}, \ldots, a_1, a_0)$ the actual computation can be accelerated. This method is summarized in Algorithm 1. The *Multiply* step of

Algorithm 1 Fixed-base comb method ([10, 6])

Precompute:

Input: Window width w, $d = \lceil t/w \rceil$, $P \in E(\mathbb{F}_q)$.
Output: Table T holding $[a_{w-1}, \ldots, a_0]_2 P$ for all length w bitstrings $(a_{w-1}, \ldots, a_0)$.

Multiply:

Input: Window width w, $d = \lceil t/w \rceil$, $k = (k_{t-1}, \ldots, k_1, k_0)_2$, lookup table T generated via *Precompute* with respect to $P \in E(\mathbb{F}_q)$.
Output: kP.

1: If necessary, pad k on the left with 0s, interpret k as $K_{w-1} \| \cdots \| K_1 \| K_0$, where each K_j is a length d bitstring, where $K_{j,i}$ denotes the i-th bit of K_j.
2: $Q = \mathcal{O}$.
3: **for** $i = d - 1$ **downto** 0 **do**
4: $\quad Q = 2Q$
5: $\quad Q = Q + [K_{w-1,i}, \ldots, K_{1,i}, K_{0,i}]_2 P = Q + T_{\sum_{j=0}^{w-1} K_{j,i} 2^j}$
6: **end for**
7: **return** Q

Algorithm 1 has an expected running time of

$$\left(\frac{2^w - 1}{2^w} d - 1\right) \mathtt{A} + (d-1)\mathtt{D}, \tag{4}$$

where A and D stand for the costs of a (mixed) point addition and a point doubling, respectively. Inside the main loop, the fixed-base comb method has the same number of point additions and point doublings. To reduce the number of point doublings by half, the so-called fixed-base comb method with two tables has been introduced, which makes use of an additional table of the same size as the first table. This method gives a benefit over the conventional comb method whenever w is chosen such that

$$\frac{2^{w-1}(w-2)}{2^w - w - 1} \geq \frac{\mathtt{A}}{\mathtt{D}}, \tag{5}$$

For instance, in case of López-Dahab coordinates, $w \geq 6$ must hold for the two-table comb method to be faster than the comb method, since $\mathtt{A}/\mathtt{D} \approx 2$ [6].

2.2 Koblitz Curves

Binary Koblitz curves are defined by Equation (2), where the curve equation is defined over $\mathbb{F}_2$, that is $a_2 \in \{0, 1\}$ and $a_6 = 1$. The group of these curves is defined over $\mathbb{F}_{2^m}$ with m prime. For cryptographic purposes, $E(\mathbb{F}_{2^m})$ is required to have almost prime group order, meaning that $E(\mathbb{F}_{2^m}) = hn$, where n is a large prime and the cofactor $h \leq 4$.

On Koblitz curves, the map $\tau : E(\mathbb{F}_{2^m}) \to E(\mathbb{F}_{2^m})$ with:

$$P \mapsto \begin{cases} (x^2, y^2) & \text{if } P = (x, y), \text{ and} \\ \mathcal{O} & \text{if } P = \mathcal{O}, \end{cases}$$

is an endomorphism, which is called *Frobenius endomorphism*. It can be used to achieve tremendous speedups for scalar multiplications on this curve type. Note that all of the remaining techniques in this section are being discussed in detail by Solinas [18].

The characteristic polynomial of τ is $\tau^2 - \mu\tau + 2$ with $\mu = (-1)^{1-a_2}$. Thus, the relation $2P = \mu\tau(P) - \tau^2(P)$ holds for all $P \in E(\mathbb{F}_{2^m})$. Since squaring is inexpensive in binary fields, the costs of applying τ are insignificant compared to other operations, such as point doubling, which suggests trading point doublings for applications of τ within multiplications by some scalar k. This way, one can trade 3M + 5S required for a doubling on Koblitz curves in López-Dahab coordinates ([9, 4, 7]) for 3S required for one evaluation of τ, where M stands for the costs of one multiplication and S for the costs of one squaring. However, to be able to do so, k needs to be converted to a value $k' = \sum_{i=0}^{l-1} u_i \tau^i \in \mathbb{Z}[\tau]$, which is said to be in τ-adic representation.

The τ-adic Representation: Now, in order to convert scalars from base 2 to base τ, we need to introduce the norm function on $\mathbb{Z}[\tau]$. The norm of an element $\alpha = a_0 + a_1\tau \in \mathbb{Z}[\tau]$ (note that every element of $\mathbb{Z}[\tau]$ can be written this way) is the integer product of α and its complex conjugate $\bar{\alpha} = a_0 + a_1\bar{\tau} \in \mathbb{Z}[\tau]$, i.e., $N : \mathbb{Z}[\tau] \to \mathbb{Z}$ with $N(a_0 + a_1\tau) = a_0^2 + \mu a_0 a_1 + 2a_1^2$. The ring $\mathbb{Z}[\tau]$ is Euclidean with respect to $N(\cdot)$, i.e., it is possible to perform division with remainder in $\mathbb{Z}[\tau]$ so that the norm of the remainder is smaller than the norm of the divisor. In particular, an element $\alpha = a_0 + a_1\tau \in \mathbb{Z}[\tau]$ is divisible by τ if and only if a_0 is even. If so, then $\alpha/\tau = (a_1 + \mu a_0/2) - (a_0/2)\tau$. Note that this also enables us to repeatedly divide $k \in \mathbb{Z}$ by τ, which is necessary to derive the τ-adic representation $k' = \sum_{i=0}^{l-1} u_i\tau^i$ of a scalar k. As $N(\tau) = 2$, the remainder in the i-th division u_i is either an element of $\{0, \pm 1\}$ in case of signed representations or an element of $\{0, 1\}$ in case of unsigned representations.

So far, there is, however, one problem with this representation, namely, the length of the resulting τ-adic representation is twice the length of the corresponding base 2 representation. In case of the NAF, this means that replacing NAF(k) by the τ-adic NAF (also known as TNAF) will eliminate one doubling for two applications of τ, but will double the number of additions in the scalar multiplication method. To overcome this problem, Solinas introduced the reduced τ-adic NAF in [18], leading to a τ-adic NAF representation half the length of the original representation and, therefore, of roughly the size of an ordinary NAF representation. Here, an element $\rho \in \mathbb{Z}[\tau]$ of smallest possible norm so that $\rho \equiv k \mod \delta$ with $\delta = \frac{\tau^m - 1}{\tau - 1}$ is computed. Then, for all points $P \in E(\mathbb{F}_{2^m})[n]$ it holds that $kP = \rho P$. From the resulting value ρ, the TNAF representation TNAF(ρ) is derived, where the length of TNAF(ρ) is bounded by $m + a_2$.

In practice, often a partial reduction (cf. [18]) is being performed in order to avoid multiprecision integer divisions. Here, one computes an element $\rho' \in \mathbb{Z}[\tau]$ that is congruent to k modulo δ, but does not necessarily have the smallest possible norm. After the partial reduction, the length of TNAF(ρ') is bounded by $m + a_2 + 3$ and is, in general, the same as the length of TNAF(ρ).

The Windowed τ-adic NAF Method: The windowed τ-adic NAF or WTNAF method is the τ-adic analogue of the ordinary width-w NAF (WNAF). The goal of this method, is to replace doublings by evaluations of the map τ and at the same time to reduce the number of additions through windowing. We do not discuss this method in detail here, but refer the reader to [6, 18].

As with the WNAF, a representation of the scalar, having at most one nonzero coefficient among w consecutive coefficients, is derived. Like before, a scalar $k \in \mathbb{Z}$ is (partially) reduced modulo $\delta = \frac{\tau^m - 1}{\tau - 1}$, and thereby an element $\rho \in \mathbb{Z}[\tau]$ is obtained. For $w > 1$, ρ can be expressed in the form $\rho = \sum_{i=0}^{l-1} u_i\tau^i$ where $u_{l-1} \neq 0$ and $u_i \in \{0\} \cup \{\pm\alpha_u : \alpha_u \in \pi\}$ with $\pi = \{\alpha_u \equiv u \mod \tau^w : u = 1, 3, \ldots, 2^{w-1} - 1\}$. Thus, the number of additions in the multiplication step is reduced in favor of the precomputation of the points $\alpha_u P$ for $\alpha_u \in \pi$. As in case of the fixed-base comb methods, these points can be computed beforehand for points known a priori, such as generators. In this case, the WTNAF method

has the following expected costs:

$$\frac{l}{w+1}\mathtt{A} + l\mathtt{T}, \tag{6}$$

where $\mathtt{T}$ denotes the cost of one evaluation of τ.

3 Fixed-base Comb Method on Koblitz Curves

We now show how the fixed-base comb multiplier can be modified to take advantage of the Frobenius endomorphism on Koblitz curves, i.e., how in this context doublings can be traded for evaluations of the map τ. Subsequently, we refer to this method as fixed-base τ-comb method. Both the precomputation and multiplication step benefit tremendously from this measure. See Section 4 for more details on the achieved performance improvements.

In order to combine the advantages of Koblitz curves with the fast fixed-based comb method, the scalar representation has to be modified. Obviously, it is necessary to obtain an unsigned τ-adic representation of scalar k, that is $k = \sum_{i=0}^{l-1} u_i \tau^i$, with $u_i \in \{0, 1\}$. This representation can be obtained with a small modification to the TNAF algorithm specified by Solinas in [18], which is summarized in Algorithm 2, where we only allow u_i to be 1 in Step 4. As with TNAF, the size of this representation is approximately twice the size of the ordinary binary representation. This is why we need to perform a reduction by $\delta = \frac{\tau^m - 1}{\tau - 1}$ beforehand, in order to reduce the size of the resulting representation. As we have obtained an unsigned τ-adic scalar representation, the doublings in

Algorithm 2 Computing the unsigned τ-adic representation

Input: $\kappa = r_0 + r_1\tau \in \mathbb{Z}[\tau]$
Output: κ represented as bitstring $u = (u_{t-1}, \ldots, u_1, u_0)_\tau$.
1: $i = 0$
2: **while** $(r_0 \neq 0 \vee r_1 \neq 0)$ **do**
3: **if** $r_0 \equiv 1 \mod 2$ **then**
4: $u_i = 1,\ r_0 = r_0 - 1$
5: **else**
6: $u_i = 0$
7: **end if**
8: $tmp = r_0,\ r_0 = \mu r_0, r_0 = r_1 + r_0/2,\ r_1 = -tmp/2$
9: $i = i + 1$
10: **end while**
11: **return** $(u_{i-1}, u_{i-2}, \ldots, u_1, u_0)$

the precomputation need to be replaced by evaluations of τ, i.e.,

$$[a_{w-1}, \ldots, a_2, a_1, a_0]_\tau P = a_{w-1}\tau^{(w-1)d}(P) + \cdots + a_2\tau^{2d}(P) + a_1\tau^d(P) + a_0 P.$$

Algorithm 3 shows the fixed-base τ-comb multiplier, where doublings are replaced by evaluations of the Frobenius endomorphism τ.

We note that due to the small costs for τ, the right hand-side of Equation (5) gets large, and, thus, the two-table τ-comb method does not give an advantage over Algorithm 3 for window sizes used in practice.

Algorithm 3 Fixed-base comb method on Koblitz curves

Precompute:

Input: Window width w, $d = \lceil t/w \rceil, P \in E(\mathbb{F}_q)$.
Output: Table T holding $[a_{w-1}, \ldots, a_0]_\tau P$ for all length w bitstrings $(a_{w-1}, \ldots, a_0)$.

Multiply:

Input: Window width w, $d = \lceil t/w \rceil, k = (k_{t-1}, \ldots, k_1, k_0)_\tau$, lookup table T generated via *Precompute* with respect to $P \in E(\mathbb{F}_q)$.
Output: kP.

1: If necessary, pad k on the left with 0s, interpret k as $K_{w-1} \| \cdots \| K_1 \| K_0$, where each K_j is a length d bitstring, where $K_{j,i}$ denotes the i-th bit of K_j.
2: $Q = \mathcal{O}$.
3: **for** $i = d - 1$ **downto** 0 **do**
4: $\quad Q = \tau Q$
5: $\quad Q = Q + [K_{w-1,i}, \ldots, K_{1,i}, K_{0,i}]_\tau P = Q + T_{\sum_{j=0}^{w-1} K_{j,i} 2^j}$
6: **end for**
7: **return** Q

4 Implementation Results

In this section, we are detailing the implementation of the τ-comb multiplication algorithm and contrast its performance to the reference implementations of the WTNAF method and the conventional comb method.

4.1 Implementation Details

We have implemented the NIST-recommended elliptic curves `K-163`, `K-233`, `K-283`, `K-409` and `K-571` [5] in Java. We are using fast reductions for the underlying binary fields and `long`-arrays for values thereof, which are in polynomial representation. We perform squarings in linear time complexity using table lookups [16] and multiplications using the windowed left-to-right comb multiplier, which works with precomputed multiplication lookup tables and is also due to Lim and Lee (cf. [12, 6]). For field multiplications, we use windows of size $w = 4$ and cache these lookup tables for recurring intermediate values in the addition/doubling formulas. Furthermore, we use partial reductions to derive the unsigned τ-adic representations (see Section 2.2).

We use López-Dahab coordinates [11] with the fast formulas given by Lange and Doche in [9, 4] as well as Higuchi and Takagi in [7] and perform mixed

additions in all implementations. Using this coordinate type, mixed additions take 8M + 5S, where the multiplication lookup tables for two intermediate values can be used twice, lowering the costs to 6M + 2m + 5S. The evaluation of τ takes 3S and point doublings take 3M + 5S, which are necessary for the conventional comb methods.

Finally, we emphasize that the implementation of the τ-comb method is straight-forward and requires only little effort, which makes it a good alternative to other fixed-base multiplication methods.

4.2 Estimated Costs and Timings

Here, we present rough estimates for the scalar multiplication costs of the discussed scalar multiplication algorithms and the corresponding timings of our software implementations. As test platform served an Intel Core i5-2540M running Ubuntu Linux 12.10/amd64 and OpenJDK 7u15/amd64 in server mode.

Table 1 lists the costs of squarings and multiplications with already precomputed lookup tables in relation to the costs of multiplications on the test platform. Thus, on the test platform, we get at most A = 8.2M, D = 3.5M and T = 0.3M per mixed addition, per doubling and per evaluation of τ, respectively.

	$\mathbb{F}_{2^{163}}$	$\mathbb{F}_{2^{233}}$	$\mathbb{F}_{2^{283}}$	$\mathbb{F}_{2^{409}}$	$\mathbb{F}_{2^{571}}$
1S =	0.094M	0.080M	0.077M	0.061M	0.055M
1m =	0.643M	0.717M	0.761M	0.829M	0.939M

Table 1: Costs of squarings in relation to multiplications, where S denotes the cost of one squaring, M the cost of a multiplication and m the cost of a multiplication using cached precomputation tables of window size $w = 4$.

In the following, we give some rough estimates of point multiplication costs of the WTNAF and the τ-comb method based on the relative costs given in Table 1. So, for instance, on curve K-233 with $w = 7$ the WTNAF method has expected costs of 284.09M, whereas the τ-comb method has expected costs of 264.36M giving an advantage of 7.46%. For $w = 8$, the expected runtime of the WTNAF method on K-283 is 314.00M, whereas the τ-comb method requires only 283.72M resulting in a performance gain of 10.67%. For $w = 9$ and curve K-233, we have 238.45M compared to 201.45M resulting in a speed-up of 18.37% and on curve K-409 with $w = 9$ we get 400.53M compared to 365.85M giving a speed-up of 9.48%.

Table 2 compares the timings of one fixed-base scalar multiplication of the τ-comb method with the WTNAF method and the conventional comb method. As Table 2 shows, in the former case, we achieve performance improvements of up to 25% and in the latter case improvements of up to 42%.

Curve	Window size	Comb [μs]	τ-Comb [μs]	WTNAF [μs]	Speedup w.r.t Comb	Speedup w.r.t WTNAF
K-163	w=6	276.50	209.06	241.77	**1.32x**	**1.16x**
	w=7	237.03	196.51	208.73	**1.21x**	**1.06x**
	w=8	211.65	168.13	194.55	**1.26x**	**1.16x**
	w=9	192.83	152.97	187.38	**1.26x**	**1.22x**
K-233	w=6	511.96	398.87	405.49	**1.28x**	**1.02x**
	w=7	455.76	341.16	386.22	**1.34x**	**1.13x**
	w=8	383.59	299.29	356.87	**1.28x**	**1.19x**
	w=9	343.87	266.21	332.61	**1.29x**	**1.25x**
K-283	w=6	738.62	544.43	583.54	**1.36x**	**1.07x**
	w=7	653.78	489.27	527.23	**1.34x**	**1.08x**
	w=8	582.79	430.37	509.92	**1.35x**	**1.18x**
	w=9	508.49	393.47	448.91	**1.29x**	**1.14x**
K-409	w=6	1581.53	1113.97	1284.27	**1.42x**	**1.15x**
	w=7	1383.75	1024.39	1110.81	**1.35x**	**1.08x**
	w=8	1203.60	889.96	1077.14	**1.35x**	**1.21x**
	w=9	1060.99	807.13	938.07	**1.31x**	**1.16x**
K-571	w=6	3026.80	2135.81	2280.14	**1.42x**	**1.07x**
	w=7	2627.00	1903.24	2061.80	**1.38x**	**1.08x**
	w=8	2336.93	1675.88	1865.80	**1.39x**	**1.11x**
	w=9	2048.58	1581.32	1816.99	**1.30x**	**1.15x**

Table 2: Comparison of the multiplication timings of the comb, the τ-comb and the WTNAF methods.

5 Conclusions

In this paper we have presented a modification to the fixed-base comb method, which allows us to benefit from speedups available on Koblitz curves, i.e., the possibility of replacing point doublings with applications of the far more efficient Frobenius endomorphism τ. In order to do so, one has to perform scalar recoding from base 2 to an unsigned base τ representation, for which we presented the respective algorithm. We have implemented the findings in Java, which allowed us to draw a detailed performance comparison. The implementation timings showed a speedup of up to 42% for single scalar multiplications over the conventional comb method and of up to 25% over the WTNAF method. Finally, we emphasize that the implementation of the τ-comb method is straight-forward and requires only little effort. All in all, this makes it a good alternative to other fixed-base multiplication methods.

5.1 Related and Future Work

The authors of [2] showed how τ-adic representations can be optimized using telescopic sums in combination with a single point halving operation giving a speed-up of $\approx 12.5\%$ over the conventional TNAF method. Later on, this approach was refined in [3] resulting in about 25% less group operations. In [1], the authors study how the Frobenius endomorphism τ can be efficiently combined with the GLV-method by using powers of τ. This way, they are able to decompose scalars and apply interleaved point multiplication on the decomposed smaller scalars, setting new speed records using a low-level implementation. Implementing the

aforementioned approaches and drawing comparisons to the method proposed in this paper are issues for future work.

The authors of [15] proposed a new, fast fixed-base comb method, which combines the Lim-Lee and the Tsaur-Chou method [19]. Future work includes adapting this new fixed-base comb multiplier to work with the Frobenius endomorphism.

References

1. Aranha, D.F., Faz-Hernández, A., López, J., Rodríguez-Henríquez, F.: Faster implementation of scalar multiplication on Koblitz curves. In: LATINCRYPT. pp. 177–193 (2012)
2. Avanzi, R.M., Ciet, M., Sica, F.: Faster scalar multiplication on Koblitz curves combining point halving with the Frobenius endomorphism. In: Public Key Cryptography - PKC 2004, 7th International Workshop on Theory and Practice in Public Key Cryptography, Singapore, March 1-4, 2004. Lecture Notes in Computer Science, vol. 2947, pp. 28–40. Springer (2004), `http://www.iacr.org/cryptodb/archive/2004/PKC/3329/3329.pdf`
3. Avanzi, R.M., Heuberger, C., Prodinger, H.: Minimality of the hamming weight of the t-naf for koblitz curves and improved combination with point halving. In: Selected Areas in Cryptography. pp. 332–344 (2005)
4. Cohen, H., Frey, G., Avanzi, R., Doche, C., Lange, T., Nguyen, K., Vercauteren, F.: Handbook of Elliptic and Hyperelliptic Curve Cryptography. CRC Press (2005)
5. Gallagher, P., Furlani, C.: FIPS PUB 186-3 federal information processing standards publication digital signature standard (dss) (2009)
6. Hankerson, D., Menezes, A.J., Vanstone, S.: Guide to Elliptic Curve Cryptography. Springer-Verlag New York, Inc., Secaucus, NJ, USA (2003)
7. Higuchi, A., Takagi, N.: A fast addition algorithm for elliptic curve arithmetic in GF(2^n) using projective coordinates. Inf. Process. Lett. 76(3), 101–103 (2000)
8. Koblitz, N.: Elliptic curve cryptosystems. Mathematics of Computation 48(177), pp. 203–209 (1987)
9. Lange, T.: A note on López-Dahab coordinates. IACR Cryptology ePrint Archive 2004, 323 (2004)
10. Lim, C.H., Lee, P.J.: More flexible exponentiation with precomputation. In: CRYPTO. pp. 95–107 (1994)
11. López, J., Dahab, R.: Improved algorithms for elliptic curve arithmetic in GF(2^n). In: Selected Areas in Cryptography. pp. 201–212 (1998)
12. López, J., Dahab, R.: High-speed software multiplication in F_{2^m}. In: INDOCRYPT. pp. 203–212 (2000)
13. Menezes, A., Okamoto, T., Vanstone, S.: Reducing elliptic curve logarithms to logarithms in a finite field. In: Proceedings of the twenty-third annual ACM symposium on Theory of computing. pp. 80–89. STOC '91, ACM, New York, NY, USA (1991)
14. Miller, V.S.: Use of elliptic curves in cryptography. In: Lecture notes in computer sciences; 218 on Advances in cryptology—CRYPTO 85. pp. 417–426. Springer-Verlag New York, Inc., New York, NY, USA (1986)
15. Mohamed, N.A.F., Hashim, M.H.A., Hutter, M.: Improved fixed-base comb method for fast scalar multiplication. In: AFRICACRYPT. pp. 342–359 (2012)

16. Schroeppel, R., Orman, H.K., O'Malley, S.W., Spatscheck, O.: Fast key exchange with elliptic curve systems. In: CRYPTO. pp. 43–56 (1995)
17. Silverman, J.: The Arithmetic of Elliptic Curves, Graduate Texts in Mathematics, vol. 106. Springer (1986)
18. Solinas, J.A.: Efficient arithmetic on Koblitz curves. Des. Codes Cryptography 19(2/3), 195–249 (2000)
19. Tsaur, W.J., Chou, C.H.: Efficient algorithms for speeding up the computations of elliptic curve cryptosystems. Applied Mathematics and Computation 168(2), 1045 – 1064 (2005), http://www.sciencedirect.com/science/article/pii/S0096300304006629