Faster and Lower Memory Scalar Multiplication on Supersingular Curves in Characteristic Three

Roberto Avanzi[1] [*] and Clemens Heuberger[2] [**]

[1] Faculty of Mathematics and Horst Görtz Institute for IT Security
Ruhr-University Bochum, Germany
`roberto.avanzi AT ruhr-uni-bochum.de`
[2] Institut für Mathematik B, Technische Universität Graz, Austria
`clemens.heuberger AT tugraz.at`

Abstract. We describe new algorithms for performing scalar multiplication on supersingular elliptic curves in characteristic three. These curves can be used in pairing-based cryptography. Since in pairing-based protocols besides pairing computations also scalar multiplications are required, and the performance of the latter is not negligible, improving it is clearly important as well. The techniques presented here bring noticeable speed ups (up to 30% for methods using a variable amount of memory and up to 46.7% for methods with a small, fixed memory usage), while at the same time bringing substantial memory reductions – factors like 3 to 8 are not uncommon.
The starting point for our methods is a structure theorem for unit groups of residue classes of a quadratic order associated to the Frobenius endomorphism of the considered curves. This allows us to define new digit sets whose elements are products of powers of certain generators of said groups. There are of course several choices for these generators: we chose generators associated to endomorphisms for which we could find efficient explicit formulae in a suitable coordinate system. A multiple-base-like scalar multiplication algorithm making use of these digits and these formulae brings the claimed speed up.

Keywords: Supersingular elliptic curves, pairing-friendly elliptic curves, scalar multiplication, Frobenius expansion, explicit formulae.

1 Introduction

The following elliptic curves over fields of characteristic three

$$\mathcal{E}_{3,\mu} : Y^2 = X^3 - X - \mu \qquad \text{with } \mu \in \{\pm 1\} \tag{1}$$

[*] Supported by the European Commission through the IST Programme under contract ICT-2007-216676 ECRYPT II.

[**] Supported by the Austrian Science Foundation FWF, project S9606, that is part of the Austrian National Research Network "Analytic Combinatorics and Probabilistic Number Theory."

are supersingular with embedding degree 6. They thus offer less security per bit than ordinary curves for DL-based applications. However, in 1998 Koblitz [20] studied their arithmetic and found their performance to be competitive with other public-key cryptosystems even after the security parameters were adjusted accordingly. Recently, pairing-based cryptography has revived the interest in these curves (together with all other types of pairing-friendly curves [16]). Most of the current research is devoted to the optimization of the field arithmetic and the pairing operation (just a few examples are [1, 9, 7, 8]).

The performance gap between pairing operations and scalar multiplication has steadily decreased in the last years. In some cases the two operations have comparable performance: In [1] variable base point scalar multiplication takes between a half and a third of the time for a Tate pairing; in [9] a η_T pairing over $\mathbb{F}_{3^{97}}$ is computed in 678 field multiplications, whereas a scalar multiplication requires, with current state-of-the art methods, at least 230 multiplications using normal bases and 296 using polynomial bases (cf. Tables 1 and 2 on page 18 under the columns labeled "BMX", that correspond to the current state of the art [10]); one of the fastest implementations of characteristic three arithmetic and of pairings in general, due to Mitsunari [22], requires 0.181 μsec, resp. 0.149 μsec for a field multiplication, resp. an η_T pairing (single threaded) over $\mathbb{F}_{3^{97}}$ – with our operation counts for [10] we extrapolate a timing of about 53.5 μsec for a scalar multiplication; over the field $\mathbb{F}_{3^{193}}$ the same implementation takes 0.624 μsec, resp. 975 μsec for the two operations – similarly we extrapolate 314.2 μsec for a scalar multiplication in this case.

For other characteristics, in [12] extensive experiments are reported not only for trace-zero varieties but also for elliptic curves, and in many cases a η_T pairing evaluation is even faster than a scalar multiplication on the same curve.

Now, most pairing-based protocols – for instance Direct Anonymous Attestation [11] – also require scalar multiplications, the latter being often performed in computationally restricted environments, such as TPM modules. Since in these cases the pairings are usually computed on faster architectures, the question of the performance of scalar multiplication becomes more, not less, severe. Thus it is an important question whether one can either speed up this operation and/or reduce its memory requirements. This is the problem studied in this paper: *We show how to perform scalar multiplication on supersingular Koblitz curves in characteristic three with extremely reduced memory requirements with respect to the state of the art while attaining better performance.*

Usually, scalar multiplication on supersingular Koblitz curves in characteristic three is done by a base three expansion [18] or by an expansion to the base of τ [20, 10], where τ is a complex number associated to the *Frobenius* endomorphism of the curve. Double base methods have been suggested in this context, for instance in [2], but only with the rational bases 2 and 3.

For the characteristic two case, it has been suggested to use the so-called *Verschiebung* endomorphism besides τ, as in [3, 4]. This is the endomorphism associated to the algebraic conjugate $\bar{\tau}$ of τ. In characteristic two τ and $\bar{\tau}$ can be used as a double bases, but, as we shall see in the next section, this cannot extended to the case considered here.

Our goal to reduce memory requirements and speeding-up scalar multiplication on the curves (1)*, is attained by finding alternative endomorphisms that can serve as additional bases to be used beside* τ*.*

This is essentially done in Section 2 by Theorem 1, that describes the structure of the group of units in the ring of residue classes of $\mathbb{Z}[\tau]$ modulo powers of the prime ideal generated by τ; the generators of this group are chosen in such a way that they can be implemented efficiently, as explained in Section 4. Their usage in practice is described in Sections 3 and 5. The techniques presented here bring noticeable speed ups – from a few percent to a near halving of computational time – while often bringing substantial memory reductions at the same time: reducing the memory requirement to just a third or even one eighth of the memory usage of the current state of the art is not uncommon. Detailed comparisons can be found in Section 5 and concluding remarks are in Section 6.

2 Digit Sets and the Structure of the Unit Group

We begin by recalling some facts on the Frobenius endomorphism τ and the associated ring $\mathbb{Z}[\tau]$.

For cryptographic applications one works in the group $\mathcal{E}_{3,\mu}(\mathbb{F}_{3^m})$ of the $\mathbb{F}_{3^m}$-rational points of the curve $\mathcal{E}_{3,\mu}$, where m is an integer not divisible by either 2 or 3, and it is usually assumed that $\mathcal{E}_{3,\mu}(\mathbb{F}_{3^m})$ contains a unique large subgroup G of prime order p. Cryptographic computations then take place in G.

The Frobenius endomorphism

$$\tau \,:\, \mathcal{E}_{3,\mu}(\mathbb{F}_{3^m}) \to \mathcal{E}_{3,\mu}(\mathbb{F}_{3^m}) \ , \quad (x,y) \mapsto (x^3, y^3)$$

is such that $\tau^2(P) - 3\,\mu\,\tau(P) + 3 \cdot P = 0$ for all points P on the curve; in other words, the relation

$$\tau^2 - 3\,\mu\,\tau + 3 = 0 \tag{2}$$

holds in the endomorphism ring of the curve. Prompted by this, we can identify τ with a imaginary quadratic solution of the last equation

$$\frac{3\mu + \sqrt{-3}}{2} \ , \tag{3}$$

which we also call τ. This induces an isomorphism (of rings with unit) between the ring $\mathbb{Z}[\tau]$ and the endomorphism ring of $\mathcal{E}_{3,\mu}$, which also maps 1 to the identity map on the curve and any rational integer n to the multiplication-by-n isogeny. This endomorphism allows for a fast scalar multiplication based on expansions of scalars to the base of τ.

We recall that $\mathbb{Z}[\tau]$ is a factorial ring (it is a quadratic order of class number 1). Also, since τ is prime, an element of $\mathbb{Z}[\tau]$ is coprime to τ if and only if τ does not divide it. Let

$$\zeta := \frac{1 - \mu\sqrt{-3}}{2} \ , \tag{4}$$

such that ζ is a primitive sixth root of unity. The complex conjugate of τ will be denoted by $\bar{\tau}$, and $\bar{\tau} = \zeta\tau$ holds. These numbers act as follows as endomorphisms on the curve:

$$\begin{aligned} \zeta &: (x, y) \mapsto (x + \mu, -y) \ , \\ \bar{\tau} &: (x, y) \mapsto (x^3 + \mu, -y^3) \ . \end{aligned}$$

Since $3 = \tau\bar{\tau} = \zeta\tau^2$, tripling is an efficient operation as well:

$$3 : (x, y) \mapsto (x^9 + \mu, -y^9) \ .$$

We see that τ and $\bar{\tau}$ are conjugated in the sense that their ratio is a unit, hence they are essentially the same base and they cannot be used in a double-base or in a double-loop scalar multiplication methods such as those from [2, 4].

Now we consider the construction of useful digit sets using the algebraic numbers we have just defined.

Similarly to Solinas [24], one can take one representative from each residue class modulo τ^w which is relatively prime to τ (a *reduced residue system* modulo τ) together with the zero to form a digit set $\mathcal{D}$ for expansions of integers $z \in \mathbb{Z}[\tau]$:

$$z = \sum_{i=0}^{\ell} d_i \tau^i \ . \tag{5}$$

If, for a given scalar z, such an expansion exists, we can use it to design a scalar multiplication method on $\mathcal{E}_{3,\mu}(\mathbb{F}_{3^m})$: First, we precompute and store all elements of the form $d \cdot P$ for $P \in \mathcal{E}_{3,\mu}(\mathbb{F}_{3^m})$, and then we can compute $z \cdot P$ by a Horner scheme:

$$\tau\Big(\tau\Big(\cdots \tau\Big(\tau\big(\tau(d_\ell P) + d_{\ell-1} P\big) + d_{\ell-2} P\Big) + \cdots + d_2 P\Big) + d_1 P\Big) + d_0 P \ .$$

An important desirable property for expansions (5) is that each block of w consecutive digits $d_i, d_{i+1}, ..., d_{i+w-1}$ contains at most one non-zero. Such an expansion is called a $\mathcal{D}$-w-NAF and a digit set $\mathcal{D}$ is called a w-Non-Adjacent-Digit-Set, or w-NADS, if every integer $z \in \mathbb{Z}[\tau]$ admits a $\mathcal{D}$-w-NAF. In analogy to the characteristic two case (cf. Solinas [24]), we can choose the elements of the digit set to be of minimal norm in their residue classes modulo τ^w. However, this only guarantees the existence of expansions for $w \geqslant 2$. Smart [23] works in a more general setting with, essentially, $w = 1$ and smallest rational integer digits (and works in general odd characteristic): in characteristic three, in order to guarantee termination, he needs an expansion with digits $\{0, \pm 1, \pm 2\}$ where *at most one digit* is exceptionally allowed to take value 2 or -2. In fact, Blake, Kumar Murty and Xu [10], observed that $\{0, \pm 1\}$ is not a 1-NADS: Indeed, if we try to expand ζ we obtain the infinite expansion

$$\zeta = -1 - \mu\tau - \tau^2 - \mu\tau^3 - \tau^4 - \mu\tau^5 - \ldots \ .$$

In what follows we shall therefore assume $w \geqslant 2$. Blake, Kumar Murty and Xu proved that a digit set of minimal norm representatives is a w-NADS. One of Koblitz' results can be formulated as saying that $\{0\} \cup \langle\zeta\rangle$ is a 2-NADS.

A big difference with respect to the characteristic two case is that in a given residue class modulo τ^w an element of minimal norm is not necessarily unique (in [5] it is fully explained when this happens, but we do not need this here). However, one is not forced to take an element from *each* of the $6 \times 3^{w-2}$ residue classes coprime to τ and store all the corresponding precomputations: Blake, Kumar Murty and Xu [10, Section 4.2] use a *signed* expansion to reduce the memory requirement by a factor of 2, i.e., to 3^{w-1} points.

But, there are other ways of constructing digit sets, and their structure can be used to design alternative scalar multiplication schemes. To achieve this, we first prove a structure theorem for the unit group of $\mathbb{Z}[\tau]$ modulo τ^w, for each natural number w. In particular we shall prove that this group is the direct product of (up to) three cyclic subgroups. Clearly, taking representatives of each class in this group will yield a reduced residue system modulo τ^w. Since the digit sets we consider in this paper consist of zero and of a reduced residue system modulo τ^w, the structure theorem will allow us to construct digit sets whose elements are products of powers of three fixed elements.

The decomposition of $(\mathbb{Z}[\tau]/\tau^w\mathbb{Z}[\tau])^\times$ can also be derived, with some effort, from Nakagoshi's much more general results for generic unit groups of residue classes of orders of number fields modulo powers of arbitrary prime ideals [21] or from Halter-Koch's classification [17] for quadratic orders. However, the proof we give here is direct and much simpler, the expressions for the generators are explicit, and the generators also enjoy the property that they correspond to endomorphisms of $\mathcal{E}_{3,\mu}$ that lend themselves to efficient evaluation.

Theorem 1. *We have*

$$\left(\frac{\mathbb{Z}[\tau]}{\tau\mathbb{Z}[\tau]}\right)^\times = \langle -1\rangle \simeq \frac{\mathbb{Z}}{2\mathbb{Z}}\ ,$$

$$\left(\frac{\mathbb{Z}[\tau]}{\tau^w\mathbb{Z}[\tau]}\right)^\times = \langle\zeta\rangle \times \left\langle 1+\mu\tau^3\right\rangle \times \langle -2\rangle \simeq \frac{\mathbb{Z}}{6\mathbb{Z}} \times \frac{\mathbb{Z}}{3^{\lfloor w/2\rfloor -1}\mathbb{Z}} \times \frac{\mathbb{Z}}{3^{\lceil w/2\rceil -1}\mathbb{Z}}\ ,\ w \geqslant 2\,.$$

Here $\langle\alpha\rangle$ denotes the group generated by α in $(\mathbb{Z}[\tau]/\tau^w\mathbb{Z}[\tau])^\times$.

Remark 1. For $w = 2$, the subgroups generated by $1 + \mu\tau^3$ and -2 in the unit group $(\mathbb{Z}[\tau]/\tau^w\mathbb{Z}[\tau])^\times$ are degenerated, i.e., they are the trivial group with one element, and for $w = 3$ only $\left\langle 1+\mu\tau^3\right\rangle$ is trivial. For $w \geqslant 4$ all three factor subgroups are not trivial.

Once the structure of $(\mathbb{Z}[\tau]/\tau^w\mathbb{Z}[\tau])^\times$ is given, a digit set for integer expansions can be built in such a way that we take one element from each residue class modulo τ^w that is not divisible by τ. It suffices to take each $d = d_1d_2d_3$ with $d_1 \in \langle\zeta\rangle$, $d_2 \in \left\langle 1+\mu\tau^3\right\rangle$, and $d_3 \in \langle -2\rangle$. The resulting digit set is invariant by multiplication by ζ, i.e., under rotation of the complex plane by $\pi/3$. The structure of this digit set will be exploited in the scalar multiplication algorithms that will be introduced in Sections 3 and 5.

In order to prove Theorem 1 we need first to compute the orders of the asserted generators modulo τ^w.

Lemma 1. *Let $k \geqslant 2$, $a \in \mathbb{Z}[\tau]$ with $\tau \nmid a$ and $w \geqslant k-1$. Then*

$$\operatorname{ord}_{\tau^w}(1+a\tau^k) = 3^{\lceil (w-k)/2 \rceil} . \tag{6}$$

In particular, we have

(a) $\operatorname{ord}_{\tau^w}(1+\mu\tau^3) = 3^{\lfloor w/2 \rfloor - 1}$ *for* $w \geqslant 2$,
(b) $\operatorname{ord}_{\tau^w}(-2) = 3^{\lceil w/2 \rceil - 1}$ *for* $w \geqslant 1$,
(c) $\operatorname{ord}_{\tau^w}(1+\mu\tau) = 3^{\lceil w/2 \rceil - 1}$ *for* $w \geqslant 3$,
(d) $\operatorname{ord}_{\tau^w}(\zeta) = 6$ *for* $w \geqslant 2$,

where $\operatorname{ord}_{\tau^w}(\alpha)$ *denotes the order of* α *in* $(\mathbb{Z}[\tau]/\tau^w\mathbb{Z}[\tau])^\times$, *i.e., the least positive exponent* r *such that* $\alpha^r \equiv 1 \pmod{\tau^w}$.

Proof (Lemma 1). We first prove

$$(1+a\tau^k)^{3^\ell} \equiv 1 + a\zeta^\ell\tau^{k+2\ell} \pmod{\tau^{2k+2\ell}} . \tag{7}$$

by induction on $\ell \geqslant 0$. For $\ell = 0$, (7) holds trivially. Assume that

$$(1+a\tau^k)^{3^\ell} = 1 + a\zeta^\ell\tau^{k+2\ell} + b\tau^{2k+2\ell}$$

for some $b \in \mathbb{Z}[\tau]$. Then we have

$$\begin{aligned}(1+a\tau^k)^{3^{\ell+1}} &= (1 + a\zeta^\ell\tau^{k+2\ell} + b\tau^{2k+2\ell})^3 \equiv 1 + 3a\zeta^\ell\tau^{k+2\ell} \\ &\equiv 1 + a\zeta^{\ell+1}\tau^{k+2\ell+2} \pmod{\tau^{2+2k+2\ell}} ,\end{aligned}$$

as $3 = \zeta\tau^2$, which concludes the proof of (7).

We set $\ell = \lceil \frac{w-k}{2} \rceil$, which results in $k + 2(\ell-1) < w \leqslant k + 2\ell$. As $\tau \nmid a\zeta^{\ell-1}$ (note that ζ is a unit in $\mathbb{Z}[\tau]$), this leads to

$$(1+a\tau^k)^{3^\ell} \equiv 1 \pmod{\tau^w} , \qquad (1+a\tau^k)^{3^{\ell-1}} \not\equiv 1 \pmod{\tau^w} .$$

Thus $\operatorname{ord}_{\tau^w}(1+a\tau^k)$ divides 3^ℓ, but does not divide $3^{\ell-1}$. We conclude that $\operatorname{ord}_{\tau^w}(1+a\tau^k) = 3^{\lceil (w-k)/2 \rceil}$, as requested.

The assertion for the special case $1+\mu\tau^3$ follows immediately from (6) by noting that $\lceil \frac{w-1}{2} \rceil = \lfloor \frac{w}{2} \rfloor$.

In order to determine the order of -2, we note that $-2 = 1 - 3 = 1 - \zeta\tau^2$, from which the result follows immediately.

Next, we have $(1+\mu\tau)^3 = 1 + (4-2\mu\tau)\tau^4$. Thus $\operatorname{ord}_{\tau^w}((1+\mu\tau)^3) = 3^{\lceil w/2 \rceil - 2}$ for $w \geqslant 3$, which implies $\operatorname{ord}_{\tau^w}(1+\mu\tau) = 3^{\lceil w/2 \rceil - 1}$ for $w \geqslant 3$.

Finally, as ζ is a primitive sixth root of unity, we have $\zeta^6 \equiv 1 \pmod{\tau^w}$. As $\zeta^3 = -1$ and $\tau \nmid 2$, we have $\zeta^3 \not\equiv 1 \pmod{\tau^w}$. Finally, $\zeta^2 - 1 = -\mu\tau$ is divisible by τ, but not τ^2, whence $\zeta^2 \not\equiv 1 \pmod{\tau^w}$ for $w \geqslant 2$. Thus $\operatorname{ord}_{\tau^w}(\zeta) = 6$ for $w \geqslant 2$. Of course, this is also a consequence of $\mathcal{D}_2$ being a 2-NADS. ☐

We are now able to prove the theorem.

Proof (Theorem 1). We prove the assertion by induction on w. We set $\alpha_1 = \zeta$, $\alpha_2 = 1 + \mu\tau^3$ and $\alpha_3 = -2$.

For $w = 1$, there is nothing to show. As $\langle\zeta\rangle$ is known to be a reduced residue system modulo τ^2 and $\langle\alpha_2\rangle$ and $\langle\alpha_3\rangle$ are both the trivial group, we are done for $w = 2$.

Assume that the result holds for some $w \geqslant 2$. We first prove that α_1, α_2, α_3 are independent modulo τ^{w+1}. So we assume that

$$\alpha_1^{a_1}\alpha_2^{a_2}\alpha_3^{a_3} \equiv 1 \pmod{\tau^{w+1}} \tag{8}$$

for some a_j with $0 \leqslant a_j < \operatorname{ord}_{\tau^{w+1}}(\alpha_j)$ for $j \in \{1, 2, 3\}$. Reducing the relation modulo τ^w, i.e.,

$$\alpha_1^{a_1}\alpha_2^{a_2}\alpha_3^{a_3} \equiv 1 \pmod{\tau^{w}} ,$$

yields $\operatorname{ord}_{\tau^w}(\alpha_j) \mid a_j$ for all $j \in \{1, 2, 3\}$.

As $\operatorname{ord}_{\tau^{w+1}}(\alpha_1) = \operatorname{ord}_{\tau^w}(\alpha_1) = 6$ by Lemma 1, we immediately get $a_1 = 0$.

By Lemma 1 we also have

$$\begin{aligned}\operatorname{ord}_{\tau^{w+1}}(\alpha_j) &= \operatorname{ord}_{\tau^w}(\alpha_j)\\ \operatorname{ord}_{\tau^{w+1}}(\alpha_k) &= 3 \cdot \operatorname{ord}_{\tau^w}(\alpha_k)\end{aligned}$$

where $\{j, k\} = \{2, 3\}$, the appropriate permutation depending on the parity of w. More precisely, it is $(j, k) = (2, 3)$ for even w and $(j, k) = (3, 2)$ for odd w.

Thus we also have $a_j = 0$ and (8) reduces to $\alpha_k^{a_k} \equiv 1 \pmod{\tau^{w+1}}$, which immediately implies $a_k = 0$, too. This concludes the proof of the independence of α_1, α_2, α_3 modulo τ^{w+1}.

As

$$|\langle\alpha_1\rangle \times \langle\alpha_2\rangle \times \langle\alpha_3\rangle| = 6 \cdot 3^{\lfloor (w+1)/2\rfloor - 1} \cdot 3^{\lceil (w+1)/2\rceil - 1} = 2 \cdot 3^w = \left|\left(\frac{\mathbb{Z}[\tau]}{\tau^w\mathbb{Z}[\tau]}\right)^{\times}\right| ,$$

the generated group has the right cardinality, so α_1, α_2, α_3 do generate the unit group. □

Remark 2. (i) For $w \geqslant 3$, the generator -2 can be replaced by $1 + \mu\tau$. This results from

$$(1 + \mu\tau) = \zeta^4(1 + \mu\tau^3)(-2)^{-1} \tag{9}$$

and $\operatorname{ord}_{\tau^w}(1 + \mu\tau) = \operatorname{ord}_{\tau^w}(-2)$ for $w \geqslant 3$.

(ii) For even values of w only,

$$(\mathbb{Z}[\tau]/\tau^w\mathbb{Z}[\tau])^{\times} = \langle\zeta\rangle \times \langle 1 + \mu\tau\rangle \times \langle -2\rangle ,$$

but not for odd w. This also follows from (9) and from the fact that $\operatorname{ord}_{\tau^w}(1 + \mu\tau) = \operatorname{ord}_{\tau^w}(-2)$. For odd w these elements generate a subgroup of index 3 of the unit group.

(iii) Further choices of generators are possible, beside -2, $1 + \mu\tau$ and $1 + \mu\tau^3$. These elements have been chosen for two reasons: (a) their relatively small norm leads to an overall well bounded norm of the digit set elements, and (b) we were able to find efficient explicit for them, cf. Section 4.

3 Scalar Multiplication Using a Factored Digit Set

Our next goal is to use the digit sets implied by the decomposition of the unit group of Theorem 1 in a precomputationless scalar multiplication algorithm similar to the one presented in [3] for Koblitz curves in characteristic two. In that case the unit group had a much simpler structure than in the present context, that will require a deeper study. From Theorem 1 we know that for $w \geqslant 2$ a decomposition $(\mathbb{Z}[\tau]/\tau^w\mathbb{Z}[\tau])^\times = \langle\zeta\rangle \times \langle\phi\rangle \times \langle\psi\rangle \simeq \mathbb{Z}/6\mathbb{Z} \times \mathbb{Z}/3^a\mathbb{Z} \times \mathbb{Z}/3^b\mathbb{Z}$ exists, where ϕ and ψ are suitable elements of $\mathbb{Z}[\tau]$ identified with the corresponding elements of the endomorphism ring of $\mathcal{E}_{3,\mu}$. Clearly, $a+b = w-2$. Assume further that we can write a scalar z in the form

$$z = \sum_{i=0}^{m} \varepsilon_i \left(\phi^{f_i}\psi^{g_i}\right)\tau^i \tag{10}$$

where $0 \leqslant f_i < 3^a$, $0 \leqslant g_i < 3^b$, and $\varepsilon_i = 0$ or $\varepsilon_i = \zeta^\ell$ with $0 \leqslant \ell < 6$. This is a τ-adic expansion where the digit set is *factored* as the product of three subgroups $\langle\zeta\rangle$, $\langle\phi\rangle$ and $\langle\psi\rangle$.

We can thus perform a scalar multiplication by means of scalar multiplication Algorithm 1 on the next page, whose correctness is an easy fact, as it simply consists of three nested Horner schemes, the two outer ones looping on the exponents of ϕ and ψ, the inner one on the exponents of τ. Note that for $w = 2$ we must have $a = b = 0$ and for $w = 3$ one of a, b must be zero (cf. Remark 1), in which cases the algorithm simplifies because there will be less nested loops.

By an easy generalization of Koblitz' arguments (cf. the end of the proof of Theorem 1 in [20]) it can be proved that the expected density of a w-NAF expansion is $2/(2w+1)$. Note that these arguments do not apply exclusively to expansions using minimal norm digits.

The expansion (10) can also be viewed as a *triple base* representation of the scalar z. Double base representations have been already considered for supersingular Koblitz curves, see for instance [2], where the possibility of using the bases 2 and 3 is mentioned but not analyzed, and in particular a rotational symmetry of the digit set, such as the one that we exploit in our algorithms, is not present.

From Theorem 1 and Remark 2 we know that we can take ζ, -2 and, depending on the parity of w, either $1+\mu\tau$ or $1+\mu\tau^3$ to generate a reduced residue set modulo τ^w, and thus a digit set $\mathcal{D}_w$. In order to guarantee that any τ-adic expansion terminates, we follow the same approach as in [3, 4], which consists in reducing the value of the parameter w if the norm of the input becomes too small. As in [3, 4] it is easy to verify that this has a minimal and asymptotically negligible impact on the weight of the expansion.

In the next section we consider the efficient implementation of the endomorphisms associated to the ring elements -2, $1+\mu\tau$ and $1+\mu\tau^3$. Since the innermost loops of Algorithm 1 are performed more often, it is a good idea to place the most expensive operation in the outermost loop and the computationally cheapest one in the innermost loop – in other words ψ should be less expensive than ϕ. In Section 5 we shall reconsider Algorithm 1 and some variants, and estimate the costs of scalar multiplication.

Algorithm 1. Low-memory τ-adic Scalar Multiplication on Koblitz Curves

INPUT: $P = (x, y) \in \mathcal{E}_{3,\mu}(\mathbb{F}_{3^m})$, scalar z represented as in Equation (10)
OUTPUT: zP

```
1.  $Q \leftarrow 0$
2.  for $j = 3^a - 1$ to $0$ do
3.      $Q \leftarrow \phi Q$                                   [skip first time]
4.      $R \leftarrow 0$
5.      for $k = 3^b - 1$ to $0$ do
6.          $R \leftarrow \psi R$                               [skip first time]
7.          $S \leftarrow 0$
8.          for $i = m - 1$ to $0$ do
9.              $S \leftarrow \tau S$                           [skip first time]
10.             if ($\varepsilon_i \neq 0$ and $f_i = j$ and $g_i = k$) then
11.                 let $\varepsilon_i = \zeta^\ell$ with $0 \leqslant \ell \leqslant 5$
12.                 switch $\ell$
13.                     case $0$: $S \leftarrow S + (x, y)$
14.                     case $1$: $S \leftarrow S + (x + \mu, -y)$
15.                     case $2$: $S \leftarrow S + (x - \mu, y)$
16.                     case $3$: $S \leftarrow S + (x, -y)$
17.                     case $4$: $S \leftarrow S + (x + \mu, y)$
18.                     case $5$: $S \leftarrow S + (x - \mu, -y)$
19.         $R \leftarrow R + S$
20.     $Q \leftarrow Q + R$
21. return $Q$
```

4 Group Operations on the Curve

In this section we show how to evaluate $1 + \mu\tau$ efficiently when the curve $\mathcal{E}_{3,\mu}$ is represented using different coordinate systems. We could not find an optimized evaluation of $1 + \mu\tau^3$, that is, just adding P and $\mu\tau^3(P)$ together seems to be the most efficient way of evaluating this endomorphism.

The costs of the various operations on the curve in different coordinate systems are compared § 4.5, in order to choose the best coordinate system for our scalar multiplication algorithms.

4.1 Explicit formulae for $(1 + \mu\tau)$ in affine coordinates

In this subsection as well as in the following three we discuss how to explicitly compute the image of a point P on $\mathcal{E}_{3,\mu}(\mathbb{F}_{3^m}) \backslash \mathcal{E}_{3,\mu}(\mathbb{F}_3)$ under the endomorphism $(1 + \mu\tau)P$ using different coordinate systems. We begin with affine coordinates.

For three points $P_i := (x_i, y_i)$, $i = 1, 2, 3$ on the curve, in the case that $P_1 \neq \pm P_2$, the expression $P_1 + P_2 = P_3$ holds with

$$x_3 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)^2 - (x_1 + x_2) \quad \text{and} \quad y_3 = y_1 + y_2 - \left(\frac{y_2 - y_1}{x_2 - x_1}\right)^3 \ . \tag{11}$$

If we take $P_2 = \mu\tau P_1$, then $(x_2, y_2) = (x_1^3, \mu y_1^3)$. For simplicity let us now omit the index 1 in what follows. We obtain

$$x_3 = \left(\frac{\mu y^3 - y}{x^3 - x}\right)^2 - (x + x^3) \quad \text{and} \quad y_3 = y + \mu y^3 - \left(\frac{\mu y^3 - y}{x^3 - x}\right)^3 \ .$$

Making use of the facts that $x^3 - x = y^2 + \mu$ and $\mu^2 = 1$ and that we are over a field of characteristic three, after a few manipulations we obtain compact expressions for x_3 and y_3,

$$x_3 = x + \mu - \frac{x^3 - x - \mu}{(x^3 - x)^2} = x + \mu - \frac{y^2}{(x^3 - x)^2} \quad \text{and} \quad y_3 = y - \frac{y^3}{(x^3 - x)^3} \ , \tag{12}$$

that will be the starting point to obtain explicit formulae in different coordinate systems.

4.2 Projective coordinates

This is a standard coordinate system for elliptic curves, where a finite point (x, y) is represented as $[X{:}Y{:}Z]$ with $x = X/Z$ and $y = Y/Z$. For the purpose of computing on supersingular elliptic curves over fields of characteristic three it has been first used by Koblitz [20]. Koblitz' formulae have been improved by Barreto, Kim, Lynn, and Scott [6].

To obtain an explicit formula consider expressions (12), first replace x, y with X/Z and Y/Z in the two formulae for x_3 and in the formula for y_3. Upon simplification, two rational expressions are obtained. Making their denominators equal and taking this value as the Z-coordinate Z_3 finally yields:

$$\begin{cases} X_3 = ((X + \mu Z)(X^3 - XZ^2)^2 - Y^2 Z^5)(X^3 - XZ^2) \ , \\ Y_3 \ = Y(X^3 - XZ^2)^3 - Y^3 Z^7 \ , \\ Z_3 \ = Z(X^3 - XZ^2)^3 \ . \end{cases}$$

This gives us the following operation sequence, where the symbols `M`, `C`, `I` shall denote a multiplication, a cubing and an inversion in $\mathbb{F}_{3^m}$. (We do not distinguish between multiplication and squaring.)

Computing $(1+\mu\tau)$ in projective coordinates Input: $[X{:}Y{:}Z]$ – Output: $[X_3{:}Y_3{:}Z_3]=(1+\mu\tau)[X{:}Y{:}Z]$		
Operation	Cost	Remark
$A \leftarrow X^3 - XZ^2$	$2\,\mathtt{M} + 1\,\mathtt{C}$	save Z^2
$B \leftarrow A^3$	$1\,\mathtt{C}$	—
$Z_3 \leftarrow ZB$	$1\,\mathtt{M}$	—
$Y_3 \leftarrow YB - (YZ^2)^3Z$	$3\,\mathtt{M} + 1\,\mathtt{C}$	save YZ^2
$X_3 \leftarrow XB + \mu Z_3 - (YZ^2)^2ZA$	$4\,\mathtt{M}$	—
Total cost:	$10\,\mathtt{M} + 3\,\mathtt{C}$	

4.3 Jacobian coordinates

These coordinates have been introduced in the context of curves in characteristic three by Harrison, Page and Smart in [18], where they are called projective, but they are long known, see for instance [13] and, for cryptographic applications [14]. In order to distinguish them from those described in § 4.2 and in accordance with the rest of the literature on elliptic curves we instead call them *Jacobian*. In Jacobian coordinates the affine point (x, y) is represented as $\langle X{:}Y{:}Z\rangle = (x, y)$, where $x = X/Z^2$ and $y = Y/Z^3$.

As in the projective case our starting point are the addition formulae (11) specialized for the case where $P_2 = \mu\tau P_1$ and simplified, i.e., (12). Putting $x = X/Z^2$ and $y = Y/Z^3$ in the formulae for x_3 and in the formula for y_3 and proceeding as above we obtain

$$\begin{cases} X_3 = ((X+\mu Z^2)(X^3 - XZ^4)^2 - Y^2Z^8)Z(X^3 - XZ^4)\ , \\ Y_3 \ = Y(X^3 - XZ^4)^3 - Y^3Z^{12}\ , \\ Z_3 \ = Z^3(X^3 - XZ^4)^3\ . \end{cases}$$

We obtain the following operation sequence:

Computing $(1+\mu\tau)$ in Jacobian coordinates Input: $\langle X{:}Y{:}Z\rangle$ – Output: $\langle X_3{:}Y_3{:}Z_3\rangle=(1+\mu\tau)\langle X{:}Y{:}Z\rangle$		
Operation	Cost	Remark
$A \leftarrow X^3 - XZ^4$	$2\,\mathtt{M} + 2\,\mathtt{C}$	save Z^4
$B \leftarrow A^3$	$1\,\mathtt{C}$	—
$Z_3 \leftarrow (ZA)^3$	$1\,\mathtt{M} + 1\,\mathtt{C}$	save ZA
$Y_3 \leftarrow YB - (YZ^4)^3$	$2\,\mathtt{M} + 1\,\mathtt{C}$	save YZ^4
$X_3 \leftarrow [(X+\mu Z^2)A^2 - (YZ^4)^2](ZA)$	$5\,\mathtt{M}$	—
Total cost:	$10\,\mathtt{M} + 5\,\mathtt{C}$	

4.4 Modified Jacobian coordinates

With these coordinates, introduced by Kim and Negre [19], an affine point (x, y) on $\mathcal{E}_{3,\mu}$ is represented by the quadruple $\langle X{:}Y{:}Z{:}T\rangle$, where $x = X/Z^2$ and $y =$

Y/Z^3 as before, and $T = Z^2$. The explicit formula in this case is obtained by modifying the formula in § 4.3.

Computing $(1+\mu\tau)$ in modified Jacobian coordinates Input: $\langle X{:}Y{:}Z{:}T\rangle$ – Output: $\langle X_3{:}Y_3{:}Z_3{:}T_3\rangle = (1+\mu\tau)\langle X{:}Y{:}Z{:}T\rangle$		
Operation	Cost	Remark
$A \leftarrow X^3 - XT^2$	$2\,\mathtt{M} + 1\,\mathtt{C}$	save T^2
$B \leftarrow A^3$	$1\,\mathtt{C}$	—
$Z_3 \leftarrow (ZA)^3$	$1\,\mathtt{M} + 1\,\mathtt{C}$	save ZA
$Y_3 \leftarrow YB - (YT^2)^3$	$2\,\mathtt{M} + 1\,\mathtt{C}$	save YT^2
$X_3 \leftarrow [(X+\mu T)A^2 - (YT^2)^2](ZA)$	$4\,\mathtt{M}$	—
$T_3 \leftarrow Z_3^2$	$1\,\mathtt{M}$	—
Total cost:	$10\,\mathtt{M} + 4\,\mathtt{C}$	

4.5 Costs of operations in different systems

We now tabulate the costs of several operations on an elliptic curve $\mathcal{E}_{3,\mu}$.

Coordinates → ↓ Operation	Affine	Projective	Jacobian	Modified Jacobian
`ADD`	$1\,\mathtt{I} + 3\,\mathtt{M}$	$14\,\mathtt{M} + 1\,\mathtt{C}$	$12\,\mathtt{M} + 4\,\mathtt{C}$	$11\,\mathtt{M} + 4\,\mathtt{C}$
`mADD`	NA	$9\,\mathtt{M} + 2\,\mathtt{C}$	$8\,\mathtt{M} + 3\,\mathtt{C}$	$7\,\mathtt{M} + 3\,\mathtt{C}$
`DBL` (also -2)	$1\,\mathtt{I} + 2\,\mathtt{M}$	$11\,\mathtt{M} + 1\,\mathtt{C}$	$7\,\mathtt{M} + 2\,\mathtt{C}$	$6\,\mathtt{M} + 4\,\mathtt{C}$
`TPL`	$4\,\mathtt{C}$	$6\,\mathtt{C}$	$1\,\mathtt{M} + 6\,\mathtt{C}$	$8\,\mathtt{C}$
τ	$2\,\mathtt{C}$	$3\,\mathtt{C}$	$3\,\mathtt{C}$	$4\,\mathtt{C}$
$1+\mu\tau$	$1\,\mathtt{I} + 2\,\mathtt{M} + 3\,\mathtt{C}$	$10\,\mathtt{M} + 3\,\mathtt{C}$	$10\,\mathtt{M} + 5\,\mathtt{C}$	$10\,\mathtt{M} + 4\,\mathtt{C}$
$1+\mu\tau^3$	$1\,\mathtt{I} + 3\,\mathtt{M} + 6\,\mathtt{C}$	$14\,\mathtt{M} + 10\,\mathtt{C}$	$12\,\mathtt{M} + 13\,\mathtt{C}$	$11\,\mathtt{M} + 16\,\mathtt{C}$

`ADD`, `DBL`, and `TPL` denote addition of two different points, doubling and tripling of a point, respectively. The prefix `m` is used to denote a *mixed* addition, i.e., addition of a point given in affine coordinates to a point in a non-affine coordinate system (in other words, $Z_2 = 1$), with a result in the same coordinate system of the second point. We did not find gains with *repeated additions*, i.e. when a given point is added to several inputs, except with standard Jacobian coordinates, where one `M` can be saved in the `ADD`. In Jacobian and modified Jacobian coordinates we save a cubing for the generic addition and nothing for the mixed addition. As symbols, τ, $1+\mu\tau$, and $1+\mu\tau^3$ denote the application of the corresponding endomorphisms.

Remark 3. (i) The costs for the projective operations have been inferred from the formulae in [20] (with some obvious simplifications) and [6].

(ii) For `mADD` in Jacobian coordinates we started with the formula for `ADD` in [18], which we report here for the reader's convenience:

$$\lambda_1 \leftarrow X_1Z_2^2\ ,\quad \lambda_2 \leftarrow X_2Z_1^2\ ,\quad \lambda_3 \leftarrow \lambda_1 - \lambda_2\ ,\quad \lambda_4 \leftarrow Y_1Z_2^3\ ,$$
$$\lambda_5 \leftarrow Y_2Z_1^3\ ,\quad \lambda_6 \leftarrow \lambda_4 - \lambda_5\ ,\quad \lambda_7 \leftarrow \lambda_1 + \lambda_2\ ,\quad \lambda_8 \leftarrow \lambda_4 + \lambda_5\ ,$$
$$Z_3 \leftarrow Z_1Z_2\lambda_3\ ,\quad X_3 \leftarrow \lambda_6^2 - \lambda_7\lambda_3^2\ ,\quad Y_3 \leftarrow \lambda_8\lambda_3^3 - \lambda_6^3\ .$$

Putting $Z_2 = 1$ we save $1\,\mathtt{M} + 1\,\mathtt{S}$ in first step, then $1\,\mathtt{M} + 1\,\mathtt{C}$ in the fourth step. Note that $\lambda_1 = X_1$ and $\lambda_4 = Y_1$, but no further savings come from this. However, in the computation of Z_3 one last $\mathtt{M}$ is saved. This brings the total cost to $8\,\mathtt{M} + 3\,\mathtt{C}$.

(iii) The formulae and relative costs for `ADD`, `DBL`, and `TPL` in modified Jacobian coordinates have been taken from [19].

Modified Jacobian coordinates are then the fastest system, as long as a field inversion is slow. In fact, according to [18] and [1], a field inversion costs more than ten field multiplications already for relatively small fields. Therefore, we shall use the modified Jacobian coordinate system in the sequel.

5 Further Improvements to Scalar Multiplication

Let us have another look at Algorithm 1: it will be the starting point for a few additional scalar multiplication methods.

Trading Frobenius Operations for Basis Conversions: If cubings are not *extremely inexpensive* or even essentially for free (such as with normal bases) they can easily become the dominant operation in Algorithm 1. In fact, there are $2 \cdot 3^w\, m$ of them.

Therefore one can envision the alternative approach of converting the coordinates of S to a normal basis representation before applying a power of τ (that in normal basis representation has nearly no computational cost) and then converting back to polynomial basis representation before adding $\zeta^\ell P$ to it. By means of this, $2\cdot 3^w m$ cubings are replaced by two basis conversion per coordinate of S for each non-zero digit, i.e., about $8 \cdot \frac{2}{2w+1}\, m$ basis conversions (assuming modified Jacobian coordinates). However, there is a more efficient approach based on the same idea: Instead of applying powers of τ to S and adding P, we convert instead the base point P to normal basis representation and apply τ^i before converting it back, applying ζ^ℓ and adding the resulting point to S. This is Algorithm 2. Only 2 basis conversions at the beginning plus 2 more for each non-zero digit in the expansion are necessary, i.e., about $2 + 2 \cdot \frac{2}{2w+1}\, m$. This is similar to the method used in [4] for elliptic Koblitz curves in characteristic two. This approach can be advantageous, but only for relatively large values of m and w, since a basis conversion can be quite expensive: Whereas for characteristic two one such operation takes about the same time as one polynomial basis multiplication [15], in characteristic three the cost can be between two and three polynomial basis multiplications.

A Different Kind of Tradeoff: The biggest advantages of Algorithms 1 and 2 lie in their minimal memory requirements, but if cubings or basis conversions are not completely negligible, performance will not be their biggest strength. However, one can exploit the structure of the unit group in a more subtle way, to store only about $O(3^{w/2})$ precomputed points instead of $O(3^w)$ to perform a width-w windowed scalar multiplication.

Algorithm 2. Low-memory τ-adic Scalar Multiplication on Koblitz Curves with basis conversions

```
INPUT: $P = (x, y) \in \mathcal{E}_{3,\mu}(\mathbb{F}_{3^m})$, scalar $z$ represented as in Equation $(10)$
OUTPUT: $zP$

1.  $Q \leftarrow 0$
2.  $\widehat{P} \leftarrow \mathrm{normal\_basis}(P)$
3.  for $j = 3^a - 1$ to $0$ do
4.      $Q \leftarrow \phi Q$                                   [skip first time]
5.      $R \leftarrow 0$
6.      for $k = 3^b - 1$ to $0$ do
7.          $R \leftarrow \psi R$                               [skip first time]
8.          $S \leftarrow 0$
9.          for $i = 0$ to $m - 1$ do
10.             if ($\varepsilon_i \neq 0$ and $f_i = j$ and $g_i = k$) then
11.                 $S \leftarrow S + \varepsilon_i \,\mathrm{polynomial\_basis}\big(\tau^i \widehat{P}\big)$
12.         $R \leftarrow R + S$
13.     $Q \leftarrow Q + R$
14. return $Q$
```

In the notation of Section 3 and in particular of Equation (10), this idea consists in (i) precomputing and storing all the points $\phi^j(P)$ for $0 \leqslant j < 3^a$, and then (ii) using a double Horner scheme on double base representation $z = \sum_{i=0}^{m} \left(\varepsilon_i \phi^{f_i}\right) \psi^{g_i} \tau^i$ with bases τ and ψ, and digits $\varepsilon_i \phi^{f_i}$, in place of the triple Horner scheme of Algorithm 1. It is clear now how to write down the methods: Algorithms 3 and 4 on page 16 are the "square root sized digit set" variants of Algorithms 1 and 2, respectively.

Comparisons: We now compare the performance and memory consumption of these algorithms to other methods presented in the scientific literature. The results are summarized in Tables 1 and 2 on page 18. We now describe our approach to the comparisons:

(i) We consider here the simple τ-adic scalar multiplication from Koblitz [20], corresponding to a τ-adic 2-NAF with digit set $\{0\} \cup \langle \zeta \rangle$, the windowed method from [10] for $w \geqslant 3$, and our four algorithms. Note that we extend the method from [10] also to $w = 5$ (since, for large m, $w = 3$ or $w = 4$ are no longer optimal). For completeness we also report the operation counts for Smart's method [23] specialized to characteristic three.

(ii) Seven different field sizes $\mathbb{F}_{3^m}$ with $m = 97$, 163, 193, 239, 509, 773 and 1223, and two representations of the fields – normal basis and polynomial basis – are considered. The first five are fields already considered in the literature, the last two have been chosen to see how the methods scale with the field size, but are not necessarily tied to particular applications.

(iii) All the computational costs are expressed in field multiplications. The cost of a field inversion is taken to be equal to 15, 17, 20, 30, 40, 60 and 80 multiplications, respectively for the seven chosen values of m, and a cubing is equal to 0.15, 0.10, 0.09, 0.07, 0.045, 0.037 and 0.033 multiplications, respectively. *Whereas using normal bases a cubing is essentially for free, the cost cannot be ignored when using a polynomial basis, because of the cost of the reduction of a polynomial of degree up to* $3(m-1)$ *modulo the defining polynomial of the field extension.* These values are approximate distillates of the values found in [18, 1] and of our own implementation experiments, and checked against Mitsunari's code [22].

(iv) For each scalar multiplication algorithm parametrized by a "window width" w, the cost corresponding to the optimal value of w is given.

(v) For all our algorithms the generators are chosen following the considerations in Remark 2: *For even* w, we take $\{\phi, \psi\} = \{-2, 1+\mu\tau\}$ and from Lemma 1 we get $a = b = 3^{w/2-1}$. *For odd* w we take $\{\phi, \psi\} = \{-2, 1+\mu\tau^3\}$; from Lemma 1 we know that $\mathrm{ord}_{\tau^w}(-2) = 3^{(w-1)/2}$, $\mathrm{ord}_{\tau^w}(1+\mu\tau^3) = 3^{(w-3)/2}$. If $\psi = -2$, then $a = (w-1)/2$, $b = (w-3)/2$, otherwise (i.e., if $\phi = -2$) $a = (w-3)/2$, $b = (w-1)/2$.
When different choices of the generators affect the performance, as in Algorithms 3 and 4, we make further case distinctions in the comparisons.

(vi) Memory consumption is given as the number of registers that are required for storing input-dependent points: The method from [10] needs to store the precomputed points other than the base point P itself, and uses an extra variable in the Horner scheme; Algorithm 1 needs to store Q, R and S and Algorithm 2 also storage $\widehat{P}$ and a copy of $\tau^i\widehat{P}$ in polynomial basis (cf. in Step 11); Algorithm 3 needs storage for the $3^a - 1$ points $\phi^j P$ with $j > 0$, as well as R, S; with respect to Algorithm 3, Algorithm 4 needs one register for P in normal basis as well, and one for the point converted in Step 7.
We do not consider the memory conversion matrices (that only apply to Algorithms 2 and 4) since they can be stored statically.

(vii) Algorithms 2 and 4 are not relevant for the normal basis comparison.

A comparison to expansions to the base of three, such as those in [18], seems due. A tripling requires twice as many cubings as a Frobenius operation. Since the density of a simple base-three expansion is 1/2 – higher than the 2/5 of Koblitz' espansion – the method is slower than Koblitz' τ-adic method. Similarly, their nonary method requiring 7 precomputations is slower than the method of Blake, Kumar, and Xu already for $w = 3$, with comparable memory requirements.

Double base chains with bases $(2, 3)$ such as those presented in [2] make sense when the doubling and tripling operation have both non-trivial costs. While computing the operation chain for a given scalar z one observes that it may end it with: (a) a doubling if $2 \mid z$; (b) a tripling if $3 \mid z$; (c) a doubling and an addition if $2 \nmid z$; or (d) a tripling and an addition/subtraction if $3 \nmid z$. The rest of the chain is the one associated to integers $z/2$, $z/3$, $(z \pm 1)/2$ and $(z \pm 1)/3$ respectively. Now, tripling in our case is always very efficient, but not doubling.

Algorithm 3. Square-root memory usage τ-adic Scalar Multiplication on Koblitz Curves

INPUT: $P = (x, y) \in \mathcal{E}_{3,\mu}(\mathbb{F}_{3^m})$, scalar z represented as in Equation (10)
OUTPUT: zP

```
1.  for $j = 0$ to $3^a - 1$ do Precompute and store $\phi^j P$
2.  $R \leftarrow 0$
3.  for $k = 3^b - 1$ to $0$ do
4.      $R \leftarrow \psi R$, $S \leftarrow 0$
5.      for $i = m - 1$ to $0$ do
6.          $S \leftarrow \tau S$                                   [skip first time]
7.          if ($\varepsilon_i \neq 0$ and $g_i = k$) then
8.              $S \leftarrow S + \varepsilon_i\big(\phi^{f_j} P\big)$
                [Use idea from Algo. 1, Steps 11–18 with $(x, y) = \phi^{f_j} P$ from table]
9.      $R \leftarrow R + S$
10. return $R$
```

Algorithm 4. Square-root memory usage Scalar Multiplication on Koblitz Curves with basis conversions

INPUT: $P = (x, y) \in \mathcal{E}_{3,\mu}(\mathbb{F}_{3^m})$, scalar z represented as in Equation (10)
OUTPUT: zP

```
1.  for $j = 0$ to $3^a - 1$ do Precompute and store $\mathrm{normal\_basis}(\phi^j P)$
2.  $R \leftarrow 0$
3.  for $k = 3^b - 1$ to $0$ do
4.      $R \leftarrow \psi R$, $S \leftarrow 0$
5.      for $i = 0$ to $m - 1$ do
6.          if ($\varepsilon_i \neq 0$ and $g_i = k$) then
7.              $S \leftarrow S + \varepsilon_i\, \mathrm{polynomial\_basis}\big(\tau^i(\phi^{f_j} P)\big)$
8.      $R \leftarrow R + S$
9.  return $R$
```

Hence, options (c) and (d) are almost always more convenient than (a) also by virtue of of the faster reduction of the intermediate results. Therefore, double base chains almost always degenerate to base-three expansions, which we have just considered.

6 Conclusions and Final Remarks

It is clear from Tables 1 and 2 on page 18 that the new methods provide a substantial improvement w.r.t. the state of the art.

1. In the case of fields represented with a polynomial basis, we see that speedups are attained already for small curves. If $w = 97$, for instance For instance, the method from [10] is already beaten by Algorithm 2 with a much lower memory usage. For $m = 509$, we obtain similar or slightly better performance using Algorithms 3 and 4, but the memory reduction goes from a factor $2.25 = 27/12$ to $6.75 = 27/4$. For even larger fields, such as $m = 1223$, the method from [10] with $w = 5$ uses 81 registers and has similar performance to Algorithm 3, but the latter uses only 10 memory registers, which therefore is about one eighth than the previous state of the art. Speed improvements are often up to 7% for variable memory usage methods (but with reduced memory usage) to 24% for methods with fixed memory usage (i.e., then comparing our first two algorithms to Koblitz' algorithm).
   We also note that whereas Algorithm 4 needs static storage for basis conversion matrices, these are not needed in Algorithm 3, that is usually just a bit slower and still faster than previous methods.
2. With a normal basis the improvements are even more impressive, going from 20% when $m = 97$ to 26% for $m = 509$ and then reaching nearly 30% for $m = 1223$, in all cases with vastly reduced memory usage.
3. If we compare methods with fixed memory consumption, we see that Algorithm 2 consistently outperforms the method of Koblitz, the speed up ranging from a few percent to 26.4% for $m = 509$ and even 46.7% for $m = 1223$ in the normal basis case (the price to pay being the usage of three intermediate registers in place of just one).

The techniques introduced in this paper therefore bring substantial speedups to scalar multiplication on supersingular Koblitz curves in characteristic three, at the same time reducing the memory footprint – by a factor roughly up to 8 in the examples we explicitly computed.
For extremely restricted environments, with no additional memory for code, the simple simple τ-adic method by Koblitz may of course still be preferable.

References

1. Omran Ahmadi, Darrel Hankerson, and Alfred Menezes, *Software Implementation of Arithmetic in $\mathbb{F}_{3^m}$*, WAIFI 2007, Springer LNCS vol. 4547, pp. 85–102. Springer, 2007.
2. Roberto Avanzi, Vassil Dimitrov, Christophe Doche, and Francesco Sica, *Extending scalar multiplication using double bases*, ASIACRYPT 2006, Springer LNCS vol. 4284, pp. 130–144. Springer, 2006,
3. Roberto Avanzi, Clemens Heuberger, and Helmut Prodinger, *On Redundant τ-adic Expansions and Non-Adjacent Digit Sets.*, SAC 2006, Springer LNCS vol. 4356, pp. 285–301. Springer, 2007
4. ———, *Redundant τ-adic Expansions I: Non-Adjacent Digit Sets and their Applications to Scalar Multiplication*, Design, Codes and Cryptography (2010), to appear.
5. ———, *Arithmetic of Koblitz Curves in Characteristic Three*, Preprint., 2010.

m	Previous methods			New methods					
	Smart [23]	Koblitz [20]	BMX [10] extended	Algo. 1	Algo. 2	Algorithm 3		Algorithm 4	
					$\psi{=}{-2}$	$\psi{=}{-2}$	$\phi{=}{-2}$	$\psi{=}{-2}$	$\phi{=}{-2}$
97	535.6	339.2	296.1	392.4	331.7	278.1	385.0	278.1	278.1
	1	1	9 (3)	3 (3)	5 (3)	4 (3)	2 (3)	6 (4)	6 (4)
163	854.4	533.5	436.5	547.5	494.2	418.5	493.3	408.8	432.0
	1	1	9 (3)	3 (3)	5 (4)	4 (3)	4 (4)	6 (4)	6 (5)
193	1001.9	624.1	503.6	621.3	567.9	483.6	553.5	468.7	492.1
	1	1	9 (3)	3 (3)	5 (4)	4 (3)	4 (4)	6 (5)	6 (5)
239	1211.8	748.7	595.5	705.3	679.2	572.5	616.9	566.0	566.0
	1	1	27 (3)	3 (3)	5 (4)	4 (3)	4 (4)	12 (6)	12 (6)
509	2509.0	1537.0	1035.4	1325.0	1300.5	1032.5	1115.0	1024.1	1024.1
	1	1	27 (4)	3 (3)	5 (5)	10 (5)	4 (4)	12 (6)	12 (6)
773	3775.2	2305.9	1528.4	1926.4	1830.4	1439.8	1597.8	1419.6	1472.9
	1	1	81 (5)	3 (3)	5 (5)	10 (5)	4 (4)	12 (6)	12 (7)
1223	5923.8	3608.0	2177.4	2930.4	2733.8	2113.5	2400.3	2111.6	2111.7
	1	1	81 (5)	3 (3)	5 (5)	10 (5)	4 (4)	30 (8)	30 (8)

Table 1. Cost – expressed in field multiplications – and random access memory usage – expressed as the number of precomputed and intermediate points – of scalar multiplication on curves over fields represented in polynomial basis. In each entry the computational cost is above, memory usage and, if applicable, the value of w between parentheses, are below.

m	Previous methods			New methods		
	Smart [23]	Koblitz [20]	BMX [10] extended	Algo. 1	Algorithm 3	
					$\psi{=}{-2}$	$\phi{=}{-2}$
97	449.2	264.6	230.0	207.1	183.9	183.9
	1	1	9 (3)	3 (3)	4 (4)	4 (4)
163	757.2	449.4	362.0	339.0	286.6	286.6
	1	1	9 (3)	3 (3)	4 (4)	4 (4)
193	897.2	533.4	410.2	388.2	333.2	316.6
	1	1	27 (4)	3 (3)	4 (4)	4 (4)
239	1111.8	662.2	487.7	466.5	399.2	375.2
	1	1	27 (4)	3 (4)	10 (5)	4 (5)
509	2371.8	1418.2	947.8	885.8	701.2	701.2
	1	1	27 (4)	3 (4)	10 (6)	10 (6)
773	3603.8	2157.4	1395.8	1246.8	985.5	985.5
	1	1	81 (5)	3 (5)	10 (6)	10 (6)
1223	5703.8	3417.4	2008.5	1819.6	1470.0	1410.0
	1	1	81 (5)	3 (5)	10 (6)	10 (7)

Table 2. Computational costs and memory consumption as in Table 1 but when using a normal basis.

6. Paulo S. L. M. Barreto, Hae Y. Kim, Ben Lynn, and Michael Scott, *Efficient algorithms for pairing-based cryptosystems*, Crypto 2002, Springer LNCS vol. 2442,

pp. 354–368. Springer, 2002.

7. Jean-Luc Beuchat, Nicolas Brisebarre, Jérémie Detrey, Eiji Okamoto, and Francisco Rodríguez-Henríquez, *A Comparison between Hardware Accelerators for the Modified Tate Pairing over F_{2^m} and F_{3^m}*, Pairing 2008, Springer LNCS vol. 5209, pp. 297–315. Springer, 2008.
8. Jean-Luc Beuchat, Emmanuel López-Trejo, Luis Martínez-Ramos, Shigeo Mitsunari, and Francisco Rodríguez-Henríquez, *Multi-core Implementation of the Tate Pairing over Supersingular Elliptic Curves*, CANS 2009, Springer LNCS vol. 5888, pp. 413–432. Springer, 2009.
9. Jean-Luc Beuchat, Masaaki Shirase, Tsuyoshi Takagi, and Eiji Okamoto, *An Algorithm for the η_T Pairing Calculation in Characteristic Three and its Hardware Implementation*, ARITH '07, pp. 97–104. IEEE Computer Society, 2007.
10. Ian F. Blake, Vijaya Kumar Murty, and Guangwu Xu, *Efficient algorithms for Koblitz curves over fields of characteristic three*, J. Discrete Algorithms **3** (2005), no. 1, pp. 113–124.
11. Ernie Brickell, Liqun Chen, and Jiangtao Li, *A New Direct Anonymous Attestation Scheme from Bilinear Maps*, Trust 2008, Springer LNCS vol. 4968, pp. 166–178. Springer, 2008.
12. Emanuele Cesena, *Trace Zero Varieties in Pairing-based Cryptography*, Ph.D. Thesis, Università degli Studi Roma TRE, 2010.
13. David V. Chudnovsky and Gregory V. Chudnovsky, *Sequences of numbers generated by addition in formal groups and new primality and factorization tests*, Advances in Applied Math. **7** (1986), 385–434.
14. Henri Cohen, Atsuko Miyaji, and Takatoshi Ono, *Efficient elliptic curve exponentiation using mixed coordinates*, Advances in Cryptology – Asiacrypt 1998, Springer LNCS vol. 1514, Springer-Verlag, Berlin, 1998, pp. 51–65.
15. Jean-Sébastien Coron, David M'Raïhi, and Christophe Tymen, *Fast generation of pairs $(k, [k]P)$ for Koblitz elliptic curves*, SAC 2001, Springer LNCS vol. 2259, pp. 151–164. Springer, 2001.
16. David Freeman, Michael Scott, and Edlyn Teske, *A Taxonomy of Pairing-Friendly Elliptic Curves*, J. Cryptology (2010), Vol 23 No. 2, pp. 224–280.
17. Franz Halter-Koch, *Einseinheitengruppen und prime Restklassengruppen in quadratischen Zahlkörpern*, Journal of Number Theory **4** (1972), pp. 10–17.
18. Keith Harrison, Dan Page, and Nigel Smart, *Software Implementation of Finite Fields of Characteristic Three, for Use in Pairing Based Cryptosystems*, LMS Journal of Computation and Mathematics **5** (2002), pp. 181–193.
19. Kwang-Ho Kim and Christophe Nègre, *Point multiplication on supersingular elliptic curves defined over fields of characteristic 2 and 3*, SECRYPT 2008, pp. 373–376. INSTICC Press, 2008.
20. Neal Koblitz, *An elliptic curve implementation of the finite field digital signature algorithm*, CRYPTO '98, Springer LNCS vol. 1462, pp. 327–337. Springer, 1998.
21. Norikata Nakagoshi, *The structure of the multiplicative group of residue classes modulo $\mathfrak{p}^{N+1}$*, Nagoya Mathematical Journal **73** (1979), pp. 41–60.
22. Shigeo Mitsunari. *A fast implementation of η_T pairing in characteristic three on intel processor.* Cryptology ePrint Archive, report 2009/032. 2009.
23. Nigel Smart, *Elliptic Curve Cryptosystems over Small Fields of Odd Characteristic.* J. Cryptology (1999) 12, pp. 141–151. Springer, 1999.
24. Jerome A. Solinas, *Efficient arithmetic on Koblitz curves*, Design, Codes and Cryptography **19** (2000), pp. 195–249. Springer, 2000.