Extending Scalar Multiplication Using Double Bases

Roberto Avanzi[1] *, Vassil Dimitrov[2],
Christophe Doche[3], and Francesco Sica[4] **

[1] Faculty of Mathematics and Horst Görtz Institute for IT Security
Ruhr-University Bochum, Germany
roberto.avanzi@ruhr-uni-bochum.de
[2] Advanced Technology Information Processing Systems laboratory,
Centre for Informations Security and Cryptography, University of Calgary, Canada
dimitrov@atips.ca
[3] Department of Computing
Macquarie University, North Ryde, NSW 2109, Australia
doche@ics.mq.edu.au
[4] Department of Mathematics and Computer Science – Acecrypt
Mount Allison University, Sackville, Canada
fsica@mta.ca – http://www.acecrypt.com

Abstract. It has been recently acknowledged [4, 6, 9] that the use of double bases representations of scalars n, that is an expression of the form $n = \sum_{e,s,t}(-1)^e A^s B^t$ can speed up significantly scalar multiplication on those elliptic curves where multiplication by one base (say B) is fast. This is the case in particular of Koblitz curves and supersingular curves, where scalar multiplication can now be achieved in $o(\log n)$ curve additions.
Previous literature dealt basically with supersingular curves (in characteristic 3, although the methods can be easily extended to arbitrary characteristic), where $A, B \in \mathbb{N}$. Only [4] attempted to provide a similar method for Koblitz curves, where at least one base must be non-real, although their method does not seem practical for cryptographic sizes (it is only asymptotic), since the constants involved are too large.
We provide here a unifying theory by proposing an alternate recoding algorithm which works in all cases with *optimal* constants. Furthermore, it can also solve the until now untreatable case where both A and B are non-real. The resulting scalar multiplication method is then compared to standard methods for Koblitz curves. It runs in less than $\log n/\log\log n$ elliptic curve additions, and is faster than any given method with similar storage requirements already on the curve K-163, with larger improvements as the size of the curve increases, surpassing 50% with respect to the τ-NAF for the curves K-409 and K-571. With respect of windowed methods, that can approach our speed but require $O(\log(n)/\log\log(n))$ precomputations for optimal parameters, we offer the advantage of a fixed, small memory footprint, as we need storage for at most two additional points.

* Partially supported by the European Commission through the IST Programme under Contract IST-2002-507932 ECRYPT.

** This work was partially supported by a NSERC Discovery Grant

1 Introduction

In cryptographic algorithms designed around elliptic curves, the most expensive part is the scalar multiplication nP, where P lies on the curve. In order to speed up this computation, it was proposed already at a very early stage of their use to adopt special families of curves where a large multiple of P can be computed very quickly. This is the case of endomorphism curves [15] or Koblitz curves E_a [17].

We will examine more closely this latter class of curves. Defined over $\mathbb{F}_{2^\mathbf{p}}$, they are endowed with the *Frobenius endomorphism* τ of the rational point group $E_a(\mathbb{F}_{2^\mathbf{p}})$. Now, τP is a large multiple of P which can be computed in time $O(1)$ using normal bases or $O(\mathbf{p})$ using polynomial bases. The map τ is also identified with a complex root of an equation of the form $\tau^2 \pm \tau + 2 = 0$ that depends only on the curve equation. Using τ, one can devise good scalar multiplication algorithms, see §§ 2.3, 2.5 and 2.6. All these algorithms compute nP with[5] $\Omega(\log n)$ costly curve operations (such as a doubling or an addition). We call these algorithms *linear* (in the number of curve operations with respect to the bit size of the field), since also the number of curve operations is $O(\log n)$. There are two ways of improving over these algorithms: either we devise algorithms with lower complexity (sublinear methods), or we reduce the number of group operations by some multiplicative factor. *We deal here with the former paradigm.*

The novelty of our approach is to combine the use of τ with double bases, first introduced in elliptic curve cryptography in [11]. To achieve this, we consider a more general setting of double base number systems (DBNS) that can be applied also to other classes of curves, such as supersingular curves over fields of characteristic 3, where in place of the Frobenius the fast operation is point tripling. We show how to find decompositions

$$n = \sum_{i=0}^{k-1} (-1)^{e_i} A^{s_i} B^{t_i}$$

with (A, B) a suitable pair of algebraic integers (such as $(2, 3)$, $(3, \tau)$, or $(\bar{\tau}, \tau)$) s_i, t_i nonnegative integers and $e_i \in \{0, 1\}$. The length k of this expansion is $O(\log n / \log\log n)$. Wo reveal, similarly to [6], a scalar multiplication algorithm with cost $O(\log n / \log\log n)$ curve operations in presence of a fast group endomorphism. We call such an algorithm *sublinear*, when the number of curve operations over the bit size of the field goes to zero.

This is a first instance of a practical sublinear scalar multiplication algorithm with very little precomputations (which depend only on $\mathbf{p}$, not the

[5] We use the notation $\Omega(x)$ to mean $> cx$ for some positive c.

curve or the point P) or storage requirements ($O(\log \mathbf{p})$ bits). We provide some computational comparisons with other methods to show that even on 163-bit curves, our method yields better results.

2 Background Material

2.1 Double Bases

Following [8], albeit with a slightly different notation, we will call a (A, B)-integer a number which can be written as $A^i B^j$ for some nonnegative integers i, j. We will extend the definition to algebraic integers, more precisely, integers in $\mathbb{Z}[\tau]$. We will also allow $A, B \in \mathbb{Z}[\tau]$. We define a (A, B)-integer expansion of n as a decomposition of n into a sum of (possibly signed) (A, B)-integers. Sometimes this will be also called a DBNS(A, B) recoding.

2.2 Koblitz Curves

For a general presentation of Koblitz curves, we refer to [13, § 15.1.1]. A Koblitz curve E_a is an elliptic curve defined over $\mathbb{F}_{2^\mathbf{p}}$, with equation

$$E_a \quad : \quad y^2 + xy = x^3 + ax^2 + 1 \ . \tag{1}$$

Here $a = 0$ or 1, and $\mathbf{p}$ is a prime chosen so to make the order of the group of points $E_a(\mathbb{F}_{2^\mathbf{p}})$ equal to twice if $a = 1$ (resp. four times if $a = 0$) a prime number, for at least one choice of a. A point $P \in E_a(\mathbb{F}_{2^\mathbf{p}})$ is then randomly chosen with order equal to that large prime. In view of Hasse's theorem, which states that $|\#E_a(\mathbb{F}_{2^\mathbf{p}}) - 2^\mathbf{p} - 1| < 2^{\frac{\mathbf{p}}{2}+1}$, this means that we can choose P so that $\operatorname{ord} P$ is very close to $2^{\mathbf{p}-1}$ if $a = 1$ and to $2^{\mathbf{p}-2}$ if $a = 0$. Since E_a has coefficients in $\mathbb{F}_2$, the Frobenius map $\tau(x, y) = (x^2, y^2)$ is an endomorphism of $E_a(\mathbb{F}_{2^\mathbf{p}})$. Since squaring is a linear operation in characteristic two, computing τP is also linear and takes time $O(\mathbf{p})$. If normal bases are used to represent elements of $\mathbb{F}_{2^\mathbf{p}}$, then computing τP is much faster, since it amounts to making two rotations, which is essentially free.

We can view τ as a complex number of norm 2 satisfying the quadratic equation $\tau^2 - (-1)^{1-a}\tau + 2 = 0$, since for any P on the curve, $\tau^2 P + 2P = (-1)^{1-a}\tau P$. Explicitly, $\tau = \frac{(-1)^{1-a}+\sqrt{-7}}{2}$. We will also make use of the conjugate $\bar{\tau} = (-1)^{1-a} - \tau$ of τ. This corresponds to the dual of the Frobenius endomorphism.

2.3 The τ-NAF for Koblitz Curves

All facts here are stated without proofs: These are found in [24, 25].

Let us consider the Koblitz curve E_a defined over $\mathbb{F}_{2^\mathbf{p}}$ by equation (1), with base point P, and let τ denote the Frobenius endomorphism. We have seen that we can view $\tau(P)$ as *multiplication by* τ and let $\mathbb{Z}[\tau]$ operate on P, but in fact there exists an integer λ such that $\tau(P) = \lambda P$, and thus τ operates on the whole subgroup generated by P like multiplication by λ.

The τ-adic non-adjacent form (τ-NAF for short) of an integer $z \in \mathbb{Z}[\tau]$ is a decomposition $z = \sum_i z_i \tau^i$ where $z_i \in \{0, \pm 1\}$ with the *non-adjacency* property $z_j z_{j+1} = 0$, similarly to the classical NAF [21]. The average *density* (that is the average ratio of non-zero bits related to the total number of bits) of a τ-NAF is $1/3$. Each integer z admits a unique τ-NAF.

The length of the τ-NAF expansion of a randomly chosen scalar n is $\approx 2\mathbf{p}$, whereas the bit length of n is $\approx \mathbf{p}$. But, for any point $P \in E_a(\mathbb{F}_{2^\mathbf{p}}) \setminus E_a(\mathbb{F}_2)$, $\tau^\mathbf{p} P = P$ and $\tau P \neq P$.

Since the ring $\mathbb{Z}[\tau]$ is Euclidean we can take the remainder ζ of n mod $\frac{\tau^\mathbf{p}-1}{\tau-1}$ and use it in place of n. This ζ will have smaller norm than that of $(\tau^\mathbf{p} - 1)/(\tau - 1)$, and thus length at most $\mathbf{p}$. Its τ-NAF is called the *reduced* τ-NAF of n and when P has prime order, it can be shown that $nP = \zeta P$.

The double-and-add scalar multiplication algorithm is a Horner scheme for the evaluation of nP using the binary expansion of $n = \sum_{i=0}^{\ell} n_i 2^i$ as $\sum_{i=0}^{\ell} n_i 2^i P$. In a similar way we can evaluate $zP = \sum_i z_i \tau^i(P)$ by a Horner scheme, and the the corresponding algorithm is called a τ-and-add algorithm. It is much faster than the double-and-add scheme on Koblitz curves because Frobenius evaluations are much faster than doublings.

2.4 Point Halving

Point halving (see [16] and [22, 23]) is a technique to improve the performance of cryptosystems based on binary elliptic curves. The idea is to replace, in the double-and-add algorithm for scalar multiplication, doublings $2Q$ by halvings $\frac{1}{2}Q = \frac{\operatorname{ord} Q+1}{2} Q$. Even though halving is not as fast as a Frobenius operation, it is much faster than doubling (between two and three times faster), according to literature [16, 22, 23] as well as [14].

2.5 Inserting a Halving in the τ-adic Scalar Multiplication

In [1] a single point halving is inserted in the "τ-and-add" scalar multiplication. This brings a non-negligible speedup (up to 14%) with respect to the use of the τ-NAF, but is not optimal. In [3] the method is refined in order to bring the speed-up to 25%, and the resulting method is proved optimal among similar methods that do not require any precomputation. The basic idea in both approaches is to express nP as $\sum_i e_{0,i} \tau^i(P) + \sum_i e_{1,i} \tau^i(Q)$ with $Q = \frac{1}{2}P$ and a smaller total Hamming weight of the $e_{j,i}$'s. The τ-and-add loop is repeated two times: first $\sum_i e_{1,i} \tau^i(P)$ is computed, then the result is halved and a second τ-and-add loop is performed like for the computation of $\sum_i e_{0,i} \tau^i(P)$, but starting with the result just obtained in place of 0.

2.6 Further Developments in τ-adic Representations

The authors of [19] generalize the approach of [1] to expressions of the form $\sum_i e_{0,i}\tau^i(P) + \sum_i e_{1,i}\tau^i(f_1(P)) + \ldots + \sum_i e_{2^{u-2}-1,i}\tau^i(f_{2^{u-2}-1}(P))$, where 1 and the f_j are representants of the residue classes modulo τ^u in the ring $\mathbb{Z}[\tau]$ which are coprime to τ, and $e_{j,i} \in \{0, \pm 1\}$. Such an expression can be obtained from a τ-adic windowed recoding [25]. If a window of width u is used, then the τ-and-add loop is performed 2^{u-2} times in place of two times as in the method of § 2.5. Thus, the number of Frobenius operations can increase exponentially with u. To ensure that this does not become a performance problem if polynomial bases are used, a technique from [20] is adopted to convert between normal and polynomial bases as required to quickly compute iterated Frobenius operations.

At the end of the τ-and-add loop corresponding to the digit f_j the partial result must be multiplied by f_{j+1}/f_j before starting the τ-and-add loop corresponding to the next digit f_{j+1}. The relations between the f_j's and their inverses must then be given explicitly. In [19] this is done for $w = 5$. Even though the authors cannot present the results in a completely general way, in the case described in [19] the reduction in memory consumption (or, equivalently, the speed-up with respect to other methods with no precomputations) is noteworthy. In order to generalize their approach the digit set itself has to be modified. In [2] it is shown how to do so.

2.7 Supersingular Elliptic Curves in Characteristic 3

We refer to [18] for generalities on supersingular elliptic curves. We will consider the curves E_b defined over $\mathbb{F}_{3^m}$ by the Weierstraß equations [5]

$$y^2 = x^3 - x + b$$

with $b = \pm 1$. On these curves, the tripling operation sends $P = (x, y)$ to $3P = (x^9 - b, -y^9)$, meaning that point tripling is essentially equivalent to two Frobenius and its cost will be considered negligible.

3 Theoretical Preliminaries

All the new results proving the sublinearity of the new DBNS decompositions are based on the following propositions. These results appears naturally in any elementary number theory book during the proof of the structure theorem for $(\mathbb{Z}/m)^\star$, the multiplicative group of invertible classes modulo m. In the sequel, we let $\mathcal{R}$ be a unique factorization domain containing $\mathbb{Z}$ (we will consider in practice $\mathcal{R} = \mathbb{Z}$ and $\mathcal{R} = \mathbb{Z}[\tau]$, where τ is the Frobenius endomorphism on a Koblitz curve). This is more stringent than necessary, however, it will make the proofs less elaborate.

Notation: For $\gcd(a,b)=1$, we denote $\mathrm{ord}_b(a)$ the multiplicative order of $a \pmod b$.

Lemma 1. *Let π be a prime, $p>0$ a generator of $\pi\mathcal{R}\cap\mathbb{Z}$ and $k\geq 2$ an integer. Let $a\in\mathcal{R}$. Then $(1+a\pi^k)^p \equiv 1+pa\pi^k \pmod{\pi^{k+2}}$.*

Proof. Note first that p is prime in $\mathbb{Z}$. Using the binomial theorem, we write out the left-hand side of the congruence as $(1+a\pi^k)^p = 1+pa\pi^k + \sum_{i=2}^{p}\binom{p}{i}a^i\pi^{ki}$. If $k\geq 2$, then $2k\geq k+2$ so that $\pi^{k+2}\mid\pi^{ki}$. □

Now the following result is proved immediately by induction.

Lemma 2. *Let π,p,k,a as in Lemma 1. If $u\geq 0$, then $(1+a\pi^k)^{p^u}\equiv 1+p^u a\pi^k \pmod{\pi^{k+u+1}}$.*

Lemma 3. *Let π,p be as in Lemma 1, $\alpha,\beta\in\mathcal{R}$ such that $\alpha\equiv\beta \pmod{\pi^u}$ for some $u\geq 1$. Then $\alpha^p\equiv\beta^p \pmod{\pi^{u+1}}$.*

Proof. We proceed as in the proof of Lemma 1. We write $\alpha=\beta+a\pi^u$. Then $\alpha^p=\beta^p+\sum_{i=1}^{p}\binom{p}{i}\beta^{p-i}a^i\pi^{iu}$. Note that $\pi^{u+1}\mid\pi^{iu}$ if $i\geq 2$. For $i=1$ the term in the summation is $p\beta^{p-1}a\pi^u$. Since $\pi\mid p$, we are done. □

Theorem 1. *Let $\alpha\in\mathcal{R}$ and $d=\mathrm{ord}_{\pi^2}(\alpha)$. Assume also that π is unramified over p, in other words that $\pi\nmid(p/\pi)$. Let*

$$k=\max\{u\geq 2\colon\, d=\mathrm{ord}_{\pi^u}(\alpha)\}\ .$$

Then

$$\mathrm{ord}_{\pi^u}(\alpha)=\begin{cases} d & \text{if } u\leq k\ ,\\ dp^{u-k} & \text{if } u>k\ .\end{cases}$$

Proof. It is clear that $\mathrm{ord}_{\pi^u}(\alpha)=d$ if $u\leq k$. We then prove by induction that

$$\mathrm{ord}_{\pi^{k+u}}(\alpha)=dp^u \quad \text{if } u\geq 1\ .$$

Since $\alpha^d\equiv 1 \pmod{\pi^k}$ we deduce by Lemma 3 $\alpha^{dp}\equiv 1 \pmod{\pi^{k+1}}$. Therefore $\mathrm{ord}_{\pi^{k+1}}(\alpha)\mid dp$ but also $d\mid\mathrm{ord}_{\pi^{k+1}}(\alpha)$ and $d\neq\mathrm{ord}_{\pi^{k+1}}(\alpha)$ by definition of u. Hence $\mathrm{ord}_{\pi^{k+1}}(\alpha)=dp$ and the initial step ($u=1$) of induction is proved.

Assume therefore that $\mathrm{ord}_{\pi^{k+u}}(\alpha)=dp^u$.

Notice also that we must then have

$$\alpha^d=1+a\pi^k \pmod{\pi^{k+1}} \quad \text{where } \pi\nmid a\ .$$

By Lemma 2, we then have

$$\alpha^{dp^u}\equiv 1+p^u a\pi^k\equiv 1+a(p/\pi)^u\pi^{k+u} \pmod{\pi^{k+u+1}}\ .$$

Since $\pi \mid p$ is unramified, we have $\alpha^{dp^u} \not\equiv 1 \pmod{\pi^{k+u+1}}$. By the induction hypothesis, $dp^u \mid \mathrm{ord}_{\pi^{k+u+1}}(\alpha)$ and we just found that these two numbers are different. Since by Lemma 3 again $\mathrm{ord}_{\pi^{k+u+1}}(\alpha) \mid dp^{u+1}$ and p is prime, it must be $\mathrm{ord}_{\pi^{k+u+1}}(\alpha) = dp^{u+1}$. This completes the proof. $\square$

We can appeal to this theorem to easily find the order of known elements to a power of a prime. We let τ be the Frobenius on a Koblitz curve as described previously, viewing it as a complex root of $X^2 + (-1)^a X + 2 = 0$. Then $\mathbb{Z}[\tau]$ is Euclidean hence a unique factorization domain. We have that τ is prime in $\mathbb{Z}[\tau]$ and likewise for $\bar{\tau} = (-1)^{a+1} - \tau$, its complex conjugate. Also, $\tau \mid 2 = \tau\bar{\tau}$ is unramified, since τ and $\bar{\tau}$ are coprime.

Corollary 1. *We have the following.*

$$\begin{aligned}
\mathrm{ord}_{3^u}(2) &= 2 \cdot 3^{u-1} && u \geq 1 \ , \\
\mathrm{ord}_{2^u}(3) &= 2^{u-2} && u \geq 3 \ , \\
\mathrm{ord}_{\tau^u}(3) &= 2^{u-2} && u \geq 3 \ , \\
\mathrm{ord}_{\tau^u}(\bar{\tau}) &= 2^{u-2} && u \geq 3 \ .
\end{aligned}$$

Proof. The first equality follows from the fact that $6 = \mathrm{ord}_9(2) < \mathrm{ord}_{27}(2)$ and an actual verification for $u = 1$.

For the second, notice that $\mathrm{ord}_4(3) = 2 = \mathrm{ord}_8(3) < \mathrm{ord}_{16}(3)$.

For the third, it suffices to notice that $2^u \mid 3^i - 1$ if and only if $\tau^u \mid 3^i - 1$. The "only if part" is obvious, since $\tau \mid 2$. For the "if" part, notice that by taking conjugates we also have $\bar{\tau}^u \mid 3^i - 1$ and since τ and $\bar{\tau}$ are coprime we get $\tau^u\bar{\tau}^u \mid 3^i - 1$.

Finally, $(-1)^{a+1}\bar{\tau} = -1 - \tau^2$, hence $\bar{\tau}^2 = 1 + \bar{\tau}\tau^3 + \tau^4$. This yields immediately $2 = \mathrm{ord}_{\tau^2}(\bar{\tau}) = \mathrm{ord}_{\tau^3}(\bar{\tau}) < \mathrm{ord}_{\tau^4}(\bar{\tau})$ if $a = 1$ or $1 = \mathrm{ord}_{\tau^2}(\bar{\tau}) < 2 = \mathrm{ord}_{\tau^3}(\bar{\tau}) < \mathrm{ord}_{\tau^4}(\bar{\tau})$ if $a = 0$ and the last formula. $\square$

This leads to the main theorem of this section.

Theorem 2. *1. Every $N \in \mathbb{Z}$ with $3 \nmid a$ is congruent modulo 3^u, $(u \geq 1)$, to precisely one of the numbers $2^j, 0 \leq j < 2 \cdot 3^{u-1}$.*

2. Every $N \in \mathbb{Z}[\tau]$ with $\tau \nmid N$ is congruent modulo τ^u, $(u \geq 3)$, to precisely one of the numbers $(-1)^e A^j, e = 0, 1$ and $0 \leq j < 2^{u-2}$, for $A = 3$ or $\bar{\tau}$.

Proof. There are exactly $\phi(3^u) = 2 \cdot 3^{u-1}$ residue classes coprime to the modulus 3^u. Hence, the first part of the theorem follows from the first equality of Corollary 1.

For the second, begin by noting that $\#\mathbb{Z}[\tau]/\tau^u = 2^u$ (since the norm of τ^u is 2^u) and $\#\big(\mathbb{Z}[\tau]/\tau^u\big)^\star = 2^{u-1}$, since elements divisible by τ are exactly the kernel of the reduction homomorphism $\mathbb{Z}[\tau]/\tau^u \to \mathbb{Z}[\tau]/\tau$. Therefore it suffices to prove that the numbers listed in the theorem are all distinct modulo τ^u. Suppose then that $(-1)^e A^j \equiv (-1)^{e'} A^{j'} \pmod{\tau^u}$. Reducing modulo τ^3, we get that $e = e'$, since the coprime residues modulo τ^3 are $\pm 1, \pm A$. Hence $A^j \equiv A^{j'} \pmod{\tau^u}$ and by Corollary 1, we must have $j = j'$. This proves the theorem. $\square$

4 Algebraic Algorithms for DBNS Recoding and Scalar Multiplication

The results hitherto proved allow us to provide new double base recodings of scalars. Unlike previous algorithms [4, 6, 8, 9] these are not greedy and proceed from right to left (i.e. from the smallest powers of the fast endomorphism to the largest).

Algorithm 1. Unsigned right-to-left DBNS(2,3) recoding

Input: An integer $n > 0$ and a parameter u.

Output: Two arrays $s[\,], t[\,]$ and their common length k. The arrays are sequences of exponents in the decomposition $n = \sum_{i=0}^{k-1} 2^{s[i]} 3^{t[i]}$

1. $N \leftarrow n$, $i \leftarrow 0$, $t \leftarrow 0$
2. $t[\,] \leftarrow 0$, $s[\,] \leftarrow 0$
3. **while** $N \geq 4^{3^{u-1}}$ **do**
4.   **while** $3 \mid N$ **do**
5. $N \leftarrow N/3$, $t \leftarrow t+1$
6.   Find $0 \leq j < 3^{u-1}2$ with $N \equiv 2^j \pmod{3^u}$
7. $N \leftarrow (N - 2^j)/3^u$
8. $s[i] \leftarrow j$, $t[i] \leftarrow t$
9. $t \leftarrow t + u$, $i \leftarrow i + 1$
10. **while** $N > 0$ **do**
11.   **while** $3 \mid N$ **do**
12. $N \leftarrow N/3$, $t \leftarrow t+1$
13.   **if** $N \equiv 1 \pmod 3$ **then**
14. $N \leftarrow (N-1)/3$, $s[i] \leftarrow 0$
15.   **else**
16. $N \leftarrow (N-2)/3$, $s[i] \leftarrow 1$
17. $t[i] \leftarrow t$, $t \leftarrow t+1$, $i \leftarrow i+1$
18. **return** $s[\,], t[\,], i$

Algorithm 1 implements a first version of a new DBNS recoding. We have given here an unsigned version, which, by a result of [4] must have at least $(1 + o(1)) \log n / \log \log n$ terms. The algorithm works by Theorem 2, which says that in Step 6 we can always find j. The termination of the algorithm is also simple here since in Step 7, N stays positive but becomes strictly smaller. A signed version, suitable for implementation on E_b, can be readily obtained and is left to the reader.

Algorithm 2 implements a signed algorithm using a complex double base $(3, \tau)$, resp. $(\bar{\tau}, \tau)$, to be used on a Koblitz curve E_a, resp. a supersingular elliptic curve in characteristic 3.

Algorithm 2. Signed right-to-left DBNS(A, τ) recoding ($A = 3$ or $\bar{\tau}$)

Input: An integer $\zeta \in \mathbb{Z}[\tau]$ and a parameter u.
Output: Three arrays $s[\,], t[\,], e[\,]$ and their common length k. The arrays are sequences of exponents in the decomposition $n = \sum_{i=0}^{k-1} (-1)^{e[i]} A^{s[i]} \tau^{t[i]}$.

```
1.  N ← ζ, i ← 0, t ← 0
2.  t[] ← 0, s[] ← 0, e[] ← 0
3.  while |N| ≥ 2^(2^(u-1))                                [See Remarks below]
4.      while τ | N do
5.          N ← N/τ, t ← t + 1
6.      Find 0 ≤ j < 2^(u-2) and e = 0, 1 with N ≡ (-1)^e A^j (mod τ^u)
7.      N ← (N - (-1)^e A^j)/τ^u
8.      s[i] ← j, t[i] ← t, e[i] ← e
9.      t ← t + u, i ← i + 1
10. while |N| > 0 do
11.     while τ | N do
12.         N ← N/τ, t ← t + 1
13.     if N ≡ 1 (mod τ^2) then
14.         N ← (N - 1)/τ^2, e[i] ← 0
15.     else
16.         N ← (N + 1)/τ^2, e[i] ← 1
17.     t[i] ← t, t ← t + 2, i ← i + 1
18. return s[], t[], e[], i
```

Remarks

1. In the case $A = \bar{\tau}$, we can replace the lower bound in line 3. by $2^{2^{u-3}}$.
2. To reduce the length of the expansion, it is possible to adapt u to the size of N. For instance, if $A = \bar{\tau}$, replace line 3. by

   3. **while** $|N| > 0$ **do**

   and after line 5. add

   6. **while** $|N| < 2^{\frac{2^{u-2}-1}{2}}$ **do** $u \leftarrow u - 1$

   Doing that, lines 10. to 17. are no longer necessary. This modification helps to save a few more additions in Algorithm 4. See Table 1.

By Theorem 2 again, the algorithm is consistent. The only point left to show is that it will terminate, namely that we have eventually $N < 2^{2^{u-1}}$, since upon entering Step 10, the algorithm computes the τ-NAF of N, hence termination is guaranteed.

Indeed notice that if $N \geq 2^{2^{u-1}}$ then

$$|(-1)^e A^j| \leq 3^j < 3^{2^{u-2}} < 4^{2^{u-2}} \leq |N| \tag{2}$$

therefore in Step 7

$$\left|\frac{N-(-1)^e A^j}{\tau^u}\right| < \frac{2\,|N|}{|\tau^u|} = \frac{|N|}{|\tau^{u-2}|} < |N| \tag{3}$$

since $u \geq 3$. Since $|N|^2 \in \mathbb{N}$ (it is the norm of the algebraic integer $N \in \mathbb{Z}[\tau]$), eventually $|N| < 2^{2^{u-1}}$ and the algorithm terminates.

In the case when $A = \bar{\tau}$ and the lower bound is $2^{2^{u-3}}$, we replace (2) by

$$|(-1)^e \bar{\tau}^j| \leq 2^{j/2} < 2^{2^{u-3}} \leq |N|$$

and we proceed as in (3) to show that $|N|$ diminishes. Therefore our algorithms are correct. Notice that we apply Algorithm 2 to ζ, the reduced τ-NAF of n (see Section 2.3).

After running Algorithms 1 or 2 and before Algorithm 3, that computes the scalar multiplication, we have to shuffle the indices i in the arrays $e[\,], s[\,], t[\,]$ so as to get $s[i+1] \geq s[i]$ for all i and $t[i+1] > t[i]$ in case $s[i+1] = s[i]$. In Algorithm 3, set $e[i] = 0$ if using an unsigned recoding.

Algorithm 3. Scalar Multiplication from a DBNS(A, B) expansion

Input: A point P on the curve E_a or E_b and the arrays $e[\,], s[\,], t[\,]$ of length k such that $s[i+1] \geq s[i]$ and $t[i+1] > t[i]$ whenever $s[i+1] = s[i]$.
Output: The point Q on E_a or E_b such that $Q = \sum_{i=0}^{k-1} (-1)^{e[i]} A^{s[i]} B^{t[i]} P$.

$Q \leftarrow \mathcal{O}$, $i \leftarrow k-1$
$s[-1] \leftarrow 0$
while $i \geq 0$ **do**
  Let $j \leq i$ be the min index with $s[j] = s[i]$
  $R \leftarrow (-1)^{e[i]} P$
  while $i > j$ **do**
    $R \leftarrow B^{t[i]-t[i-1]} R + (-1)^{e[i-1]} P$
    $i \leftarrow i - 1$
  $Q \leftarrow Q + R$
  $Q \leftarrow A^{s[i]-s[i-1]} Q$
return Q

5 Comparison with Established Methods

We want here to give an idea of how well Algorithm 2 fares with $(\bar{\tau}, \tau)$ on Koblitz curves standardized by NIST. We compare our new multiplication algorithm with the τ-and-add using a τ-NAF expansion [24] and the width-w τ-NAF expansion [25].

For a given value of u, by (3), the number of iterations in the main loop (Steps 3 to 9) is bounded by the quantity c such that $|\zeta| = |\tau^{u-2}|^c = 2^{\frac{c}{2}(u-2)}$. This gives

$$c = \frac{2\log_2|\zeta|}{u-2} = \frac{\mathbf{p}}{u-2}$$

for a generic scalar, by the way ζ is constructed. Also, since the "tail" (i.e. the quantity processed in Steps 11 to 17) is a generic integer of $\mathbb{Z}[\tau]$ of norm less than $2^{2^{u-2}}$, its expected Hamming weight is bounded by $2^{u-2}/3$. Thus, the average Hamming weight of the new expansion is bounded by

$$\frac{\mathbf{p}}{u-2} + \frac{2^{u-2}}{3} \quad ,$$

and its worst case by

$$\frac{\mathbf{p}}{u-2} + 2^{u-3} + 1 \quad . \tag{4}$$

In practice, when N is large in (3), the new value of N has absolute value much closer to $|N|/|\tau^u|$, therefore we should expect a Hamming weight closer to the value

$$\frac{\mathbf{p}}{u} + \frac{2^{u-2}}{3} \quad . \tag{5}$$

Algorithm 3 then implies that the total cost of a scalar multiplication equals at most $\mathbf{p}/u + 2^{u-2}/3$ additions plus 2^{u-2} applications of $\bar{\tau}$. Since an application of $\bar{\tau} = (-1)^{1-a} - \tau$ corresponds to a curve addition, the total cost (in curve additions) is bounded from above by

$$f(u) = \frac{\mathbf{p}}{u} + \frac{2^u}{3} \quad .$$

In the previous argument, following [4, Section 4], we neglected the cost of applying τ, as we will in the following comparisons. See also Section 7 for a concrete approach to reducing the impact of the Frobenius to a non-dominant term.

We can modify Algorithm 3 to make use of the advantage of halvings over multiplications by $A = \bar{\tau}$ (at least a 50% saving in performance). Indeed, let $\zeta' = 2^{2^{u-2}}\zeta \pmod{\frac{\tau^{\mathbf{p}}-1}{\tau-1}}$ with minimal norm. From a DBNS$(\bar{\tau}, \tau)$ expansion

$$\zeta' = \sum_{i=0}^{k-1} (-1)^{e'_i} \bar{\tau}^{s'_i} \tau^{t'_i}$$

get that

$$nP = \zeta P = \sum_{i=0}^{k-1} (-1)^{e'_i} \frac{\bar{\tau}^{s'_i}}{2^{2^u}} \tau^{t'_i} P = \sum_{i=0}^{k-1} (-1)^{e'_i} \frac{\tau^{t'_i - s'_i}}{2^{2^u - s'_i}} P$$
$$= \sum_{i=0}^{k-1} (-1)^{e'_i} \frac{\tau^{\epsilon_i \mathbf{p} + t'_i - s'_i}}{2^{2^u - s'_i}} P$$

where $\epsilon_i = 1$ if $t'_i < s'_i$ and 0 else. Note that this is a valid DBNS$(1/2, \tau)$ expansion, because for different values of i, j, the same powers of $1/2$ and τ occur only if $s'_i = s'_j$ and either $t'_i - s'_i = t'_j - s'_j$ or $t'_i - s'_i = \mathbf{p} + t'_j - s'_j$. Since the pairs (s'_i, t'_i) arise from a DBNS expansion and $t'_i < \mathbf{p}$, either case is impossible.

In this case, from (5) and the subsequent analysis, we can conclude that the cost of one scalar multiplication using a DBNS$(1/2, \tau)$ expansion is upper bounded on average by $g(u)$ curve additions, where

$$g(u) = \frac{\mathbf{p}}{u} + \frac{5}{24} 2^u \quad .$$

For various parameters of $\mathbf{p}$ corresponding to the NIST curves K-163 ($a = 1$), K-233 ($a = 0$), K-283 ($a = 0$), K-409 ($a = 0$), K-571 ($a = 0$), Table 1 gives the scalar multiplication costs in elliptic curve additions (with the assumption that two halvings are equivalent to one addition) using the τ-NAF, width-w τ-NAF (w-τ-NAF) and our new recodings, on average, as well as the percentage improvement over those methods and the value of u used in minimizing the functions $f(u)$ and $g(u)$. In each case, the average is computed over 25,000 values.

Field size $\mathbf{p}$	τ-NAF	w-τ-NAF	w	DBNS$(\bar{\tau}, \tau)$	u	DBNS$(\frac{1}{2}, \tau)$	u	%/τ-NAF	%/w-τ-NAF
163	54.33	34.16	5	34.60	5	31.09	5	42.78%	8.99%
233	77.66	45.83	5	46.60	5	41.38	6	46.72%	9.71%
283	94.33	54.16	5	54.38	5	48.80	6	48.27%	9.90%
409	136.33	73.42	6	74.40	6	66.89	6	50.94%	8.90%
571	190.33	102.37	6	97.18	6	88.04	7	53.74%	14.00%

Table 1. Comparison of scalar multiplication algorithms on Koblitz curves

6 Asymptotic Improvements

We now establish the asymptotic behavior of our new scalar multiplication algorithm. Its sublinear nature will be thus revealed. We have the following.

Theorem 3. *Algorithms 1 and 2 allow to express* nP*, where* $P \in E_b$ *or* $P \in E_a$*, as*

$$nP = \left(\sum_{i=0}^{k-1} (-1)^{e_i} A^{s_i} B^{t_i}\right) P \quad \textit{with} \quad (s_i, t_i) \neq (s_j, t_j) \quad \textit{for} \quad i \neq j \ ,$$

where $(A, B) = (2, 3)$ *in the case of* E_b *and* $(A, B) = (3, \tau)$ *or* $(\bar{\tau}, \tau)$ *in the case of* E_a*. The length* k *satisfies on average (the worst case being twice as large only in the case of* E_a*)*

$$k \leq \big(1 + o(1)\big) \frac{\log n}{\log \log n} \quad \textit{as}\, n \to \infty \ ,$$

and $\max s_i \leq \log n / (\log \log n)^2$.

Therefore scalar multiplication nP *can be performed via Algorithm 3 on these curves with an average cost of less than* $\big(1 + o(1)\big) \log n / \log \log n$ *curve additions.*

Proof. We detail the proof in the case of Koblitz curves. In the DBNS(2, 3) case, simple modifications lead to the analogous result. We start with (4), letting $u = \lfloor 2 + \log_2 \mathbf{p} - 2 \log_2 \log \mathbf{p} \rfloor$. We then find that $k \leq \frac{\mathbf{p}}{\log_2 \mathbf{p}} + o\left(\frac{\mathbf{p}}{\log \mathbf{p}}\right)$. Since on average $\mathbf{p} = \log_2 n$ we are done in the average case. In the worst case $\mathbf{p}$ has to be replaced by $2 \log_2 |\zeta|$, where $\zeta = n$ if n is too small. The (average) bound on the s_i is immediate from Step 6 in Algorithm 2.

Since the total cost of Algorithm 3 differs from the Hamming weight k by a multiple of $2^{u-2} = o(\mathbf{p} / \log \mathbf{p})$ we are done. □

7 On the Use of Normal vs. Polynomial Bases

Neglecting the cost of τ is fine if normal bases are used, but when polynomial bases are used Frobenius operations can become expensive as u increases. One solution is provided, as already mentioned, by a technique introduced by Park et al. in [20] and used by Okeya et al. in [19]. Instead of applying a variable power of the Frobenius to a changing point as done in Steps 5 to 9 if Algorithm 3, we apply the Frobenius to the point P and accumulate directly. Only, the Frobenius is performed on a copy of P that has been converted to normal basis representation (hence, all powers of the Frobenius have essentially the same cost), and then the result is converted back to polynomial basis representation before adding it to the accumulator variable that will contain the final result at the end of the algorithm.

Algorithm 4. $(\bar{\tau}, \tau)$-Double Bases Scalar Multiplication on Koblitz Curves

Input: A point P on E_a, a scalar z and arrays $e[\,], s[\,], t[\,]$ of length k with $s[i+1] \geq s[i]$ such that $z = \sum_{i=0}^{k-1} (-1)^{e[i]} \bar{\tau}^{s[i]} \tau^{t[i]}$.

Output: The point Q on E_a such that $Q = zP = \sum_{i=0}^{k-1} (-1)^{e[i]} \bar{\tau}^{s[i]} \tau^{t[i]} P$.

1. $R \leftarrow \mathrm{normal\_basis}(P)$ [Keep in affine coordinates]
2. $Q \leftarrow 0$ [Use López-Dahab coordinates]
3. **for** $i = k-1$ to 0 **do**
4.   **if** $i \neq k-1$ **and** $s[i] \neq s[i+1]$ **then**
5.   **for** j $= 1$ **to** $s[i+1] - s[i]$ **do**
6. $Q \leftarrow \tau^{-1} Q$, $Q \leftarrow 2Q$
7. $Q \leftarrow Q + e[i] \cdot \mathrm{polynomial\_basis}(\tau^{t[i]} R)$ [Mixed coordinates]
8. **return** Q

With our notation the resulting method is presented as Algorithm 4, in a version that uses mixed coordinate arithmetic and projective $(\mathcal{P})$ or López-Dahab $(\mathcal{LD})$ coordinates [12, § 13.3] while keeping the points P and R in affine $(\mathcal{A})$ coordinates.

There we use the fact that $2 = \tau\bar{\tau}$ to implement $\bar{\tau}$ as a doubling with an inverse of a Frobenius, an operation that requires three square root extractions in $\mathcal{P}$ or $\mathcal{LD}$. A square root extraction costs between 1/8 and 1/2 of a multiplication depending on the field [14]. A doubling in $\mathcal{LD}$ costs 4 multiplications and 4 squarings, whereas a mixed coordinate addition (i.e. adding a point in $\mathcal{A}$ to a point in $\mathcal{LD}$ with a result in $\mathcal{LD}$) costs 9 multiplications and 5 squarings. The time required by a basis conversion (routines normal_basis and polynomial_basis) is roughly the same as one polynomial basis multiplication, and the conversion routines require each a matrix that occupies $O(\mathbf{p}^2)$ bits of storage [7]. Hence Steps 1 and 6 cost each about two field multiplications. The time for an evaluation of $\bar{\tau}$ is then roughly a half of the time for an evaluation of the addition (including the basis conversion).

8 Conclusion

This work shows that using double bases in scalar multiplication improves performance significantly, even for the smallest cryptographic parameters, at almost no additional memory cost. This method however is only effective if multiplication by one of the bases can be neglected, as was shown in [4]. The resulting new scalar multiplication algorithms are especially fast on Koblitz curves and supersingular curves of characteristic three used in pairing-based cryptosystems.

As this work is being written, other articles on the same subject are about to be published. In [10], accepted at CHES 2006, the authors present practical measurements on FPGA and show that indeed one achieves a 50% speedup already on the smallest Koblitz curve K-163 by using short decompositions found by a clever extensive search. The paper [2], to appear in the proceedings of SAC 2006, among other things contains results similar to ours, but expressed in the language of expansions with respect to a single base using suitably defined digit sets.

References

1. R. Avanzi, M. Ciet, and F. Sica. *Faster Scalar Multiplication on Koblitz Curves combining Point Halving with the Frobenius Endomorphism.* In *Proceedings of PKC 2004*, Lecture Notes in Computer Science 2947, pp. 28–40. Springer, 2004.
2. R. Avanzi, C. Heuberger, and H. Prodinger. *On Redundant τ-adic Expansions and Non-Adjacent Digit Sets.* To appear in *Proceedings of SAC 2006 (Workshop on Selected Areas in Cryptography)*, Lecture Notes in Computer Science. Springer.
3. R. Avanzi, C. Heuberger, and H. Prodinger. *Minimality of the Hamming Weight of the τ-NAF for Koblitz Curves and Improved Combination with Point Halving.* In *Proceedings of SAC 2005*, Lecture Notes in Computer Science 3897, pp. 332–344. Springer, 2006.
4. R. Avanzi and F. Sica. *Scalar Multiplication on Koblitz Curves using Double Bases.* To appear in *Proceedings of Vietcrypt 2006*, Lecture Notes in Computer Science. Springer, 2006.
5. P. Barreto, H. Y. Kim, B. Lynn, and M. Scott. *Efficient algorithms for pairing-based cryptosystems.* In *Advances in Cryptology - CRYPTO 2002*, Lecture Notes in Computer Science 2442, pp. 354–369. Springer, 2002.
6. M. Ciet and F. Sica. *An Analysis of Double Base Number Systems and a Sublinear Scalar Multiplication Algorithm.* In *Progress in Cryptology - Proceedings of Mycrypt 2005*, Lecture Notes in Computer Science 3715, pp. 171–182. Springer, 2005.
7. J.-S. Coron, D. M'Raïhi, and C. Tymen. *Fast generation of pairs $(k, [k]P)$ for Koblitz elliptic curves.* In *Proceedings of SAC 2001*, Lecture Notes in Computer Science 2259, pp. 151–164. Springer, 2001.
8. V. S. Dimitrov, L. Imbert, and P. K. Mishra. *Efficient and secure elliptic curve point multiplication using double-base chains.* In *Advances in Cryptology - ASIACRYPT 2005*, Lecture Notes in Computer Science 3788, pp. 59–78. Springer, 2005.
9. V. S. Dimitrov, L. Imbert, and P. K. Mishra. *Fast elliptic curve point multiplication using double-base chains.* Cryptology ePrint Archive, Report 2005/069, 2005. Available from `http://eprint.iacr.org/`.
10. V. S. Dimitrov, K. Jarvinen, M. J. Jacobson Jr, W. F. Chan, and Z. Huang. *FPGA Implementation of Point Multiplication on Koblitz Curves Using Kleinian Integers.* In *Proceedings of CHES 2006*, Lecture Notes in Computer Science. Springer, 2006.
11. V. S. Dimitrov, G. A. Jullien, and W. C. Miller. *An algorithm for modular exponentiation. Information Processing Letters*, 66(3):155–159, 1998.
12. C. Doche and T. Lange. *Arithmetic of Elliptic Curves, in Handbook of Elliptic and Hyperelliptic Curve Cryptography, H. Cohen and G. Frey Eds.* CRC Press, Inc., 2005.

13. C. Doche and T. Lange. *Arithmetic of Special Curves, in Handbook of Elliptic and Hyperelliptic Curve Cryptography, H. Cohen and G. Frey Eds.* CRC Press, Inc., 2005.
14. K. Fong, D. Hankerson, J. López, and A. J. Menezes. *Field Inversion and Point Halving Revisited. IEEE Trans. Comp.*, 53(8), pp. 1047–1059, August 2004.
15. R. P. Gallant, J. L. Lambert, and S. A. Vanstone. *Faster Point Multiplication on Elliptic Curves with Efficient Endomorphisms.* In *Advances in Cryptology - CRYPTO 2001*, Lecture Notes in Computer Science 2139, pp. 190–200. Springer, 2001.
16. E. W. Knudsen. *Elliptic Scalar Multiplication Using Point Halving.* In *Advances in Cryptography - ASIACRYPT 1999*, Lecture Notes in Computer Science 1716, pp. 135–149. Springer, 1999.
17. N. Koblitz. *CM-curves with good cryptographic properties.* In *Advances in Cryptology - CRYPTO 1991*, Lecture Notes in Computer Science 576, pp. 279–287, Berlin, 1991. Springer.
18. A. J. Menezes. *Elliptic Curve Public Key Cryptosystems.* Kluwer Academic Publishers, 1993.
19. K. Okeya, T. Takagi, and C. Vuillaume. *Short Memory Scalar Multiplication on Koblitz Curves.* In *Proceedings of CHES 2005*, Lecture Notes in Computer Science 3659, pp. 91–105. Springer, 2005.
20. D. J. Park, S. G. Sim, and P. J. Lee. *Fast scalar multiplication method using change-of-basis matrix to prevent power analysis attacks on Koblitz curves.* In *Proceedings of WISA 2003*, Lecture Notes in Computer Science 2908, pp. 474–488. Springer, 2003.
21. G.W. Reitwiesner. *Binary arithmetic. Advances in Computers*, 1, pp. 231–308, 1960.
22. R. Schroeppel. *Elliptic curve point ambiguity resolution apparatus and method.* International Application Number PCT/US00/31014, filed 9 November 2000.
23. R. Schroeppel. *Elliptic curves: Twice as fast!*, 2000. Presentation at the Crypto 2000 Rump Session.
24. J. A. Solinas. *An Improved Algorithm for Arithmetic on a Family of Elliptic Curves.* In *Advances in Cryptology - CRYPTO 1997*, Lecture Notes in Computer Science 1294, pp. 357–371. Springer, 1997.
25. J. A. Solinas. *Efficient arithmetic on Koblitz curves. Designs, Codes and Cryptography*, 19, pp. 195–249, 2000.