

An Analysis of Double Base Number Systems and a Sublinear Scalar Multiplication Algorithm

Mathieu Ciet¹ and Francesco Sica^{2*}

¹ Gemplus Security Technologies Department
La Vigie, ZI Athélia IV, Av. du Jujubier,
B.P. 100, 13705 La Ciotat Cedex, France
mathieu.ciet@gemplus.com

² AceCrypt and Mount Allison University
Department of Mathematics and Computer Science
67 York Street, Sackville, NB, E4L 1E6, Canada
fsica@mta.ca – <http://www.acecrypt.com>

In memoriam Ioannis Pauli II

Abstract. In this paper we produce a practical and efficient algorithm to find a decomposition of type

$$n = \sum_{i=1}^k 2^{s_i} 3^{t_i}, \quad s_i, t_i \in \mathbb{N} \cup \{0\} \quad \text{with} \quad k \leq (c + o(1)) \frac{\log n}{\log \log n} .$$

It is conjectured that one can take $c = 2$ above. Then this decomposition is refined into an effective scalar multiplication algorithm to compute nP on some supersingular elliptic curves of characteristic 3 with running time bounded by

$$O\left(\frac{\log n}{\log \log n}\right)$$

and essentially no storage. To our knowledge, this is the first instance of a scalar multiplication algorithm that requires $o(\log n)$ curve operations on an elliptic curve over \mathbb{F}_q with $\log q \approx \log n$ and uses comparable storage as in the standard double-and-add algorithm.

This leads to an efficient algorithm very useful for cryptographic protocols based on supersingular curves. This is for example the case of the well-studied (in the past four years) identity based schemes. The method carries over to any supersingular curve of fixed characteristic.

Keywords. Integer decomposition, exponentiation algorithms.

1 Introduction

In asymmetric cryptographic algorithms, the costliest part in execution time is most often the computation of nP , for P belonging to some group¹

* This work was partially supported by a NSERC Discovery Grant

¹ This operation is called exponentiation or scalar multiplication, according to the multiplicative, respectively additive, notation of the group law.

denoted additively. The standard algorithm to perform this is to decompose n in base 2 and to apply a double-and-add algorithm. If n is randomly chosen with a fixed number of bits, then this algorithm requires on average $\log_2 n$ doublings and $(\log_2 n)/2$ additions. Hence we can say that a classical double-and-add algorithm takes time² $\sim c \log n$ (asymptotic to $c \log n$ on average) for some $c > 0$.

In some specific groups, one can improve the constant c by taking for instance signed binary expansions [10] or expansions to other bases [3, 11]. We call *linear* an algorithm with running time T satisfying $c \log n < T < d \log n$ for some $c, d > 0$. Similarly, we shall call it *sublinear* if the running time is $o(\log n)$ (little oh) as n goes to infinity. This means that this time is strictly smaller than $\epsilon \log n$ for any $\epsilon > 0$.

The aim of the present work is to present the first practical sublinear scalar multiplication algorithm. The algorithm is sublinear when used on particular supersingular elliptic curves of fixed characteristic (for instance, elliptic curves defined over \mathbb{F}_p). We shall deal with characteristic 3 here as an illustration of the method and leave the easy modifications to the interested reader.

The algorithm proceeds as follows. We first decompose the multiplier n as

$$n = \sum_{i=1}^k 2^{s_i} 3^{t_i}, \quad s_i, t_i \in \mathbb{N} \cup \{0\} \quad (1)$$

with $(s_i, t_i) \neq (s_j, t_j)$ for $i \neq j$ and

$$k \leq (c + o(1)) \frac{\log n}{\log \log n}$$

for some proven $c > 0$ (conjecturally $c = 2$). This allows us to build a double-and-add algorithm where the number of additions is $O(\frac{\log n}{\log \log n})$.

Then a simple remark allows us to impose a further condition that $\max(s_i) \leq \log^{1-\delta} n$ for any $0 < \delta < 1$. This has the effect of bringing the number of doublings down to $O(\log^{1-\delta} n)$.

We still need to worry about triplings, but on a supersingular curve these are practically as fast as two Frobenius endomorphisms, hence they can be neglected if normal bases are used and even in the case of polynomial bases, they are faster than a doubling or an addition.

Therefore the total running time of our algorithm will be bounded by the number of additions, that is $O(\frac{\log n}{\log \log n})$.

² The time unit is an elliptic curve operation, say an addition, since a doubling takes a positive proportion of performing an addition.

Let us mention that the idea of using (1), called in the literature double base number system, goes back at least to [5], where even an exponentiation algorithm is given [6] with comparable running time as ours. However this algorithm requires $O(\log^2 n)$ in storage, whereas our algorithm only needs a minimal storage. Also [4] develops these ideas into a generic scalar multiplication algorithm that would have fast running time. Unfortunately, no explicit estimate is given at the moment.

2 Notations and Mathematical Background

2.1 Notations

We define a s -integer as an integer whose prime factorization contains only the first s primes. A 2-integer will also be called a binumber. An expansion such as (1) will be called a binumber expansion of length k instead of a double base number system, to reflect the fact that the double base is $(2, 3)$.

We are interested in doing asymptotic analysis, at first. We will freely use the abusive notation $f(s) \leq g(s)$ to indicate in fact that for any $C > 1$ one has $f(s) \leq Cg(s)$ as $s \rightarrow \infty$. It must then be understood that the optimal asymptotic result will be achieved when we let $C \rightarrow 1^+$.

In the same vein, the ϵ 's are not always the same from one equation to the next, but have to be understood as an arbitrarily small constant, independent of any other quantity.

Logarithms are taken to the base e in our theoretical analysis, however this is irrelevant in the algorithms, since the ϵ in the exponents absorbs the constants involved in changing bases.

2.2 Continued Fractions

Every real number α has an essentially unique (unique for irrationals) expression as a sequence of "infinite quotients", called continued fraction expansion

$$\alpha = a_0 + \frac{1}{a_1 + \frac{1}{a_2 + \dots}}$$

where $(a_s)_{s \geq 0}$ is a sequence of integers which are all positive except $a_0 = \lfloor \alpha \rfloor$. We then write $\alpha = [a_0, a_1, \dots]$. The a_i 's are called partial quotients, and the rationals

$$\frac{p_s}{q_s} = [a_0, a_1, \dots, a_s]$$

are called the convergents to α . They provide the best rational approximation among all fractions with denominator bounded by q_s . The following recurrence relations are satisfied

$$p_s = a_s p_{s-1} + p_{s-2} \quad \text{and} \quad q_s = a_s q_{s-1} + q_{s-2} . \quad (2)$$

It is also known that the convergents hop from one side to the other of α . In other words,

$$\alpha - \frac{p_{2s}}{q_{2s}} > 0 \quad \text{and} \quad \alpha - \frac{p_{2s+1}}{q_{2s+1}} < 0 \quad (3)$$

2.3 Measure of Irrationality

The irrationality measure $\mu(\alpha)$ of $\alpha \in \mathbb{R} - \mathbb{Q}$ is defined as

$$\mu(\alpha) = \sup \left\{ r \in \mathbb{R} : \exists \infty (p, q) \in \mathbb{Z}^2 \text{ with } \left| \alpha - \frac{p}{q} \right| \leq \frac{1}{q^r} \right\} .$$

It is also known that the convergents $\frac{p}{q}$ of the continued fraction expansion of α satisfy

$$\left| \alpha - \frac{p}{q} \right| \leq \frac{1}{q^2} , \quad (4)$$

hence $\mu(\alpha) \geq 2$. The set of reals with irrationality measure greater than 2 has Lebesgue measure zero, even if it is equipotent to the reals. Therefore we should expect that a “random” number α would have $\mu(\alpha) = 2$.

2.4 Supersingular Elliptic Curves in Characteristic 3

We refer to [9] for generalities on supersingular elliptic curves. We will focus here on supersingular elliptic curves defined over \mathbb{F}_{3^p} and in particular on the examples E_b given in [1] by the Weierstraß equations

$$y^2 = x^3 - x + b \quad \text{with } b = \pm 1.$$

On these curves, the tripling operation sends $P = (x, y)$ to $3P = (x^9 - b, -y^9)$, so that point tripling is essentially equivalent to two Frobenius applications and can be done in $O(p)$ binary operations using polynomial bases and in $O(1)$ binary operations (rotations) using normal bases. As well, $3^t P$ can be computed in $O(pt)$ binary steps using polynomial bases and essentially $O(1)$ using normal bases. In any case, if $t \leq p$, this cost is less than that of multiplying elements in \mathbb{F}_{3^p} hence is less than one elliptic curve operation. This is the reason why we will neglect this cost henceforth.

3 Short Binumber Expansions

The goal of the present section is to provide an effective algorithm to compute a binumber expansion of any n of length $O(\log n / \log \log n)$, with an explicitly given constant. We will give a mathematical proof with an asymptotic value for the constant, then give a standard conjecture under which one can hope to improve this constant (effective measurements corroborate this – see [4]).

The idea of the algorithm is the use of a more explicit tool than [12], namely a consequence of the following more recent result [7, p.19]. For any $a, b \in \mathbb{N}$ with $(a + b) > e^{1000}$ we have

$$|a \log 3 - b \log 2| \geq \exp(-153500) \max(a, b)^{-40499} \quad (5)$$

In particular, this implies that

$$\mu \left(\frac{\log 2}{\log 3} \right) \leq 40500 \quad (6)$$

Let us denote $\mu = \mu \left(\frac{\log 2}{\log 3} \right)$ and $\frac{p_s}{q_s}$ the sequence of the convergents to $\log_3 2$. Also as before, we let $(a_i)_{i \geq 0}$ be its sequence of partial quotients. Then, choosing any small $\epsilon > 0$, as s becomes large,

$$\frac{1}{q_s^{\mu+\epsilon}} \leq \left| \frac{\log 2}{\log 3} - \frac{p_s}{q_s} \right| < \frac{1}{a_{s+1} q_s^2} . \quad (7)$$

This implies that $a_s \leq q_{s-1}^{\mu-2+\epsilon}$. In view of (2) one has

$$q_s = a_s q_{s-1} + q_{s-2} \leq q_{s-1}^{\mu-1+\epsilon} \quad (8)$$

Let $m > 1$ be a real parameter to be fixed later and let s be henceforth chosen as the smallest even index such that $q_s > m$.

Lemma 1. *Using the above notations we have*

$$\exp \left(\frac{1}{m^{(\mu-1)^3+\epsilon}} \right) < \frac{2^{q_s}}{3^{p_s}} < \exp \left(\frac{1}{m^{\mu-1-\epsilon}} \right) . \quad (9)$$

Proof. By (8) and the definition of s we deduce that

$$q_s \leq m^{(\mu-1)^2+\epsilon} . \quad (10)$$

Together with (3) and the left-hand inequality in (7) we get

$$q_s \frac{\log 2}{\log 3} - p_s > \frac{1}{m^{(\mu-1)^3+\epsilon}}$$

and so

$$q_s \log 2 - p_s \log 3 > \frac{1}{m^{(\mu-1)^3+\epsilon}}$$

which is the left-hand side of (9). The right-hand side is then a consequence of the definition of μ , since

$$q_s \frac{\log 2}{\log 3} - p_s \leq \frac{1}{q_s^{\mu-1-\epsilon}} < \frac{1}{m^{\mu-1-\epsilon}} .$$

□

Remark 1. Note that in the preceding equation, the ϵ can be dropped by (4) in the probable case that $\mu = 2$.

Theorem 1. *Let n be a large integer. There exists a binumber N satisfying*

$$n - \frac{n}{\log^{\frac{1}{\mu(\mu-1)}-\epsilon} n} \leq N \leq n$$

Remark 2. It is of course possible to give a totally effective version of this theorem, valid for all values of n , but this would clutter the proof with useless technicalities. In practice we see that the range of validity of such inequalities starts at small values of n .

Proof. Our idea is to start from the largest power 3^ν not exceeding n and then to multiply it by the t -th power of $\frac{2^{q_s}}{3^{p_s}}$ which by (9) is very close to 1. However one must be careful of not dividing by too large a power of 3. We will then get a binumber that is very close to n , provided we choose the largest t satisfying both

$$\left(\frac{2^{q_s}}{3^{p_s}} \right)^t \leq \frac{n}{3^\nu} \tag{11}$$

and

$$p_s t < \nu . \tag{12}$$

The former inequality, together with (9), implies that

$$t \leq m^{(\mu-1)^3+\epsilon} \tag{13}$$

since $n/3^\nu \leq 3$. Inequality (12) can be automatically verified if m is not too large. Note that (7) and (10) imply that

$$\frac{q_s}{2} < p_s < q_s < m^{(\mu-1)^2+\epsilon} . \tag{14}$$

This implies that

$$p_s t \leq m^{(\mu-1)^2+\epsilon} m^{(\mu-1)^3+\epsilon} < \nu = \left\lfloor \frac{\log n}{\log 3} \right\rfloor ,$$

if we let

$$m = (\log n)^{\frac{1}{\mu(\mu-1)^2-\epsilon}} . \quad (15)$$

We will henceforth fix m to this value. The choice of t is then dictated by (11) alone and is

$$t = \left\lfloor \frac{\log n - \left\lfloor \frac{\log n}{\log 3} \right\rfloor \log 3}{q_s \log 2 - p_s \log 3} \right\rfloor = \left\lfloor \frac{\log_3 n - \lfloor \log_3 n \rfloor}{q_s \log_3 2 - p_s} \right\rfloor . \quad (16)$$

Let $N = 3^\nu \left(\frac{2^{q_s}}{3^{p_s}}\right)^t$. Then N is a binumber less than n and using (9) we get

$$\frac{n}{N} < \exp\left(\frac{1}{m^{\mu-1-\epsilon}}\right) \implies N > n \left(1 - \frac{1}{m^{\mu-1-\epsilon}}\right)$$

which in view of (15) proves the left-hand side inequality of the theorem. \square

Theorem 2. *Denote the measure of irrationality of $\log 2/\log 3$ by μ . Then every sufficiently large number n can be written as a sum*

$$n = \sum_{i=1}^k 2^{s_i} 3^{t_i}, \quad s_i, t_i \in \mathbb{N} \cup \{0\}$$

with

$$k \leq \mu(\mu-1) \frac{\log n}{\log \log n} + o\left(\frac{\log n}{\log \log n}\right) .$$

Proof. Apply Theorem 1 iteratively, starting from $\mathbf{n} = n$, finding N and replacing \mathbf{n} with $\mathbf{n} - N$. At each step, the size of \mathbf{n} is divided by a factor $\log^{\frac{1}{\mu(\mu-1)-\epsilon}} \mathbf{n}$. Therefore, κ steps at most are needed to shrink the size of \mathbf{n} from n to \sqrt{n} , where

$$\left(\log^{\frac{1}{\mu(\mu-1)-\epsilon}} \sqrt{n}\right)^\kappa = \sqrt{n} .$$

This yields

$$\kappa = \left(\frac{\mu(\mu-1)}{2} + \epsilon\right) \frac{\log n}{\log \log \sqrt{n}} .$$

Repeating this process, replacing n by \sqrt{n} , we see that the number of iterations needed to get down from n to e^4 is bounded by

$$\begin{aligned} & \left(\frac{\mu(\mu-1)}{2} + \epsilon \right) \sum_{0 \leq u \leq \frac{\log \log n}{\log 2} - 2} \frac{\log \sqrt[2^u]{n}}{\log \log \sqrt[2^{u+1}]{n}} = \\ & \left(\frac{\mu(\mu-1)}{2} + \epsilon \right) \sum_{0 \leq u \leq \frac{\log \log n}{\log 2} - 2} \frac{1}{2^u} \frac{\log n}{\log \log n - (u+1) \log 2} . \end{aligned}$$

Upon splitting the last sum into subsums for $0 \leq u < \frac{3 \log \log \log n}{\log 2}$ and $\frac{3 \log \log \log n}{\log 2} \leq u \leq \frac{\log \log n}{\log 2} - 2$ we get on the former sum a bound of

$$\begin{aligned} & \left(\frac{\mu(\mu-1)}{2} + \epsilon \right) \log n \sum_{u=0}^{\infty} \frac{1}{2^u} \frac{1}{\log \log n - 4 \log \log \log n} \\ & \leq (\mu(\mu-1) + \epsilon) \frac{\log n}{\log \log n} , \end{aligned}$$

while on the latter a smaller bound

$$O \left(\frac{\log n}{2^{\frac{3 \log \log \log n}{\log 2}} \log \log n} \right) = O \left(\frac{\log n}{(\log \log n)^2} \right) .$$

This proves the theorem. □

Remark 3. Note that in the preceding theorem one can assume $2^{s_i} 3^{t_i} \neq 2^{s_j} 3^{t_j}$ if $i \neq j$. In fact by (12) we make sure N in Theorem 1 will not be a pure power of 2, as soon as n is sufficiently large. Below this threshold we can represent any number in base 2 as a sum of $O(1)$ powers of 2, so it suffices to show that $2^{s_{i+1}} 3^{t_{i+1}} < 2^{s_i} 3^{t_i}$ when these numbers are large. Let \mathbf{n} then be so large that $\log^{\frac{1}{\mu(\mu-1)-\epsilon}} \mathbf{n} > 2$. Following the constructive proof of the preceding theorem, we have

$$2^{s_{i+1}} 3^{t_{i+1}} \leq \mathbf{n} - 2^{s_i} 3^{t_i} \leq \frac{\mathbf{n}}{\log^{\frac{1}{\mu(\mu-1)-\epsilon}} \mathbf{n}} < \mathbf{n} - \frac{\mathbf{n}}{\log^{\frac{1}{\mu(\mu-1)-\epsilon}} \mathbf{n}} \leq 2^{s_i} 3^{t_i} .$$

In practice \mathbf{n} does not have to be large, since it is expected that $\mu = 2$.

The proofs of Theorem 1 and 2 show the workings of an algorithm that allows us to find a binumber expansion of n , given as Algorithm 1. The choice for m in step 5 was done to fix ideas (we also assume in

Input: An integer $n \geq 1$

Output: A set $\mathcal{S} = \{(s_1, t_1), \dots, (s_k, t_k)\}$ with the property that $n = \sum_{i=1}^k 2^{s_i} 3^{t_i}$ with s_i, t_i nonnegative integers

1. $i \leftarrow 0, \mathcal{S} = \emptyset$
 2. While $n > 0$ do
 3. $i \leftarrow i + 1$
 4. $\nu \leftarrow \lfloor \log_3 n \rfloor$
 5. $m \leftarrow \lfloor \sqrt[3]{\nu} \rfloor$
 6. $s \leftarrow \min\{2j : q_{2j} > m\}$
 7. $q \leftarrow q_s, p \leftarrow p_s$
 8. $t \leftarrow \left\lfloor \frac{\log_3 n - \lfloor \log_3 n \rfloor}{q \log_3 2 - p} \right\rfloor$
 9. $N \leftarrow 2^{qt} 3^{\nu - pt}$
 10. $\mathcal{S} \leftarrow \mathcal{S} \cup \{(qt, \nu - pt)\}, n \leftarrow n - N$
 11. Return \mathcal{S}
-

Algorithm 1. Binumber Chain

implementations that $\mu = 2$) and because this algorithm is to be used in conjunction with Algorithm 2, where some extra conditions, which are satisfied here, are imposed on the binumber expansion.

Note that the storage requirement is minimal and only depends on the size of the size of n . If storing all p_s, q_s with $\max_s(p_s, q_s) \leq M$, we need only $O(\log^2 M)$ bits, since (2) implies $2q_{s-2} < q_s$ and similarly for p_s . Hence we would need only $O((\log \log n)^2)$ bits of storage.

4 Scalar Multiplication on Supersingular Elliptic Curves in Characteristic 3

It appears that the decomposition described above leads quite naturally in some cases to sublinear multiplication algorithms, that is to algorithms employing $o(p)$ elliptic curve operations to compute nP on an elliptic curve defined over \mathbb{F}_q with $q = 2^p, 3^p$. Of these two types, ternary curves (defined over \mathbb{F}_{3^p}) seem to be of some interest in conjunction with recent works on pairing-based cryptographic protocols [2, 8]. It happens that examples of supersingular curves over finite fields of characteristic 3 exhibit the right ingredient to make our multiplication algorithm run provably fast. Namely, such elliptic curves possess an extremely fast multiplication by 3 (comparable to the Frobenius endomorphism).

Our previous algorithm does not immediately generalize to a fast multiplication algorithm, since the cost of the latter is expressed in terms of curve additions and doublings/triplings. Theorem 2 results in a sublinear number of curve additions, but, if a doubling or a tripling is performed in a non-negligible fraction of the time cost of an addition, then the overall cost of the scalar multiplication is still comparable to $\log n$. In the case of a supersingular elliptic curve in characteristic 3, we will however prove the following result.

Theorem 3. *Let E be a supersingular elliptic curve defined over \mathbb{F}_3 and $P \in E(\mathbb{F}_{3^m})$. Then there exists an algorithm which, on input $n < \#E(\mathbb{F}_{3^m})$, will compute nP in time*

$$O\left(\frac{\log n}{\log \log n}\right) .$$

Proof. Let us decompose n as in the proof of Theorem 1. Then (13) and (14) imply that

$$q_s t \leq \log^{1-\delta} n \tag{17}$$

as soon as

$$m = (\log n)^{\frac{1}{\mu(\mu-1)^2-\varepsilon}}$$

for some fixed $\varepsilon > 0$. Note that in Algorithm 1 our choice for m leads to (17) with $\delta = 1/3$. But $q_s t$ defines an exponent s_i at each step of Algorithm 1. The conclusion is that we can efficiently decompose

$$n = \sum_{i=1}^I 2^{s_i} 3^{t_i}, \quad s_i, t_i \in \mathbb{N} \cup \{0\} \tag{18}$$

with

$$I \leq \mu(\mu-1) \frac{\log n}{\log \log n} + o\left(\frac{\log n}{\log \log n}\right) \tag{19a}$$

and the additional conditions (see Remark 3)

$$(s_i, t_i) \neq (s_j, t_j), \quad i \neq j \tag{19b}$$

and

$$\max_i s_i \leq \log^{1-\delta} n . \tag{19c}$$

From a decomposition with these properties, it is now easy to build a scalar multiplication algorithm that takes asymptotically (as we take $\delta \rightarrow 0$) at most $(\mu(\mu-1) + o(1)) \frac{\log n}{\log \log n}$ curve operations (additions or doublings) to compute nP , as described below.

□

Note that (19b) implies that we can rewrite (18) as³

$$n = \sum_{i=1}^{\mathcal{J}} 2^{s_i} \sum_{j=1}^{\mathcal{J}_i} 3^{t_{i,j}} \quad (20)$$

where

$$\sum_{i=1}^{\mathcal{J}} \sum_{j=1}^{\mathcal{J}_i} 1 = I, \quad s_i > s_{i+1} \quad \text{and} \quad t_{i,j} > t_{i,j+1} .$$

This, together with (19c), implies that

$$\mathcal{J} \leq \log^{1-\delta} n . \quad (21)$$

Algorithm 2 describes a practical use of (20) in a scalar multiplication algorithm employing two nested loops, one (internal) on the index j , another (external) on the index i . The total cost of the algorithm is bounded by

$$\sum_{i=1}^{\mathcal{J}} \sum_{j=1}^{\mathcal{J}_i} \text{additions} + \mathcal{J} \text{ doublings} = O\left(\frac{\log n}{\log \log n}\right) \text{ elliptic curve operations}$$

using (19a) and (21). Note that we can neglect the cost of performing $3^{t_{i,j}-t_{i,j+1}}R$, following the discussion in Section 2.4: if we use normal bases, this is clear since the cost of the triplings in steps 5 and 10 is negligible. If we use a suitable polynomial basis then the cost of triplings from those steps is $O(pt_{i,1} + pt_{\mathcal{J},1}) = O(p^2)$ binary operations. Since this is less than a curve operation, the total cost of triplings is, after (21), $O(\log^{1-\delta} n)$ elliptic curve operations, so that this is negligible with respect to the total number of additions.

5 Conclusion and Ideas for Further Research

We have presented a scalar multiplication algorithm on supersingular elliptic curves (in characteristic 3, easily extendible to other small characteristics) which is sublinear in the length of the multiplier. Its relative speedup over double-and-add or even triple-and-add in cryptographic algorithms making use of these elliptic curves increases with the size of the parameters towards 100%. This means that larger parameter size will result in comparatively more and more advantageous performance with this sublinear algorithm.

³ The s_i 's here are a subsequence of the original s_i 's.

Input: A point P on the supersingular elliptic curve E_b and a sequence of pairs of exponents $(s_i, t_{i,j})$ as in (20).

Output: The point Q on the elliptic curve such that $Q = nP$.

1. $Q \leftarrow \mathcal{O}$
 2. For $i = 1$ to $J - 1$
 3. $R \leftarrow P$
 4. For $j = 1$ to \mathcal{J}_i
 5. $R \leftarrow 3^{t_{i,j} - t_{i,j+1}} R + P$
 6. $Q \leftarrow Q + R$
 7. $Q \leftarrow 2^{s_i - s_{i+1}} Q$
 8. $R \leftarrow P$
 9. For $j = 1$ to \mathcal{J}_J
 10. $R \leftarrow 3^{t_{J,j} - t_{J,j+1}} R + P$
 11. $Q \leftarrow Q + R$
 12. Return Q
-

Algorithm 2. Sublinear Multiplication

From another viewpoint, this sheds new light on the balance between weakened security on such curves and added performance: in increasing the size of the parameters, one would still retain a high performance. In conclusion choosing these curves (with sufficient large parameters for good security) in pairing-based cryptography seems a viable choice that needs further investigation.

References

1. P. Barreto, H.Y. Kim, B. Lynn, and M. Scott. Efficient algorithms for pairing-based cryptosystems. In M. Yung, editor, *Advances in Cryptology - Proceedings of CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 354–369. Springer, 2002.
2. D. Boneh, B. Lynn, and H. Shacham. Short signatures from the Weil pairing. *Journal of Cryptology*, 17(4):297–319, September 2004.
3. M. Ciet, T. Lange, F. Sica, and J-J. Quisquater. Improved algorithms for efficient arithmetic on elliptic curves using fast endomorphisms. In E. Biham, editor, *Advances in Cryptology - Proceedings of Eurocrypt 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 388–400. Springer, 2003.
4. 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. <http://eprint.iacr.org/>.

5. V. S. Dimitrov, G. A. Jullien, and W. C. Miller. Theory and applications for a double-base number system. In *IEEE Symposium on Computer Arithmetic*, pages 44–53, 1997.
6. V. S. Dimitrov, G. A. Jullien, and W. C. Miller. An algorithm for modular exponentiation. *Information Processing Letters*, 66(3):155–159, 1998.
7. N. Gouillon. *Minorations explicites de formes linéaires en deux logarithmes*. PhD thesis, Université de la Méditerranée Aix-Marseille II, Faculté des Sciences de Luminy, 2003.
8. A. Joux. A one round protocol for tripartite Diffie-Hellman. In W. Bosma, editor, *Algorithmic Number Theory, 4th International Symposium, ANTS-IV*, volume 1838 of *Lecture Notes in Computer Science*, pages 385–394. Springer, 2000.
9. A.J. Menezes. *Elliptic Curve Public Key Cryptosystems*. Kluwer Academic Publishers, 1993.
10. F. Morain and J. Olivos. Speeding up the Computations on an Elliptic Curve using Addition-Subtraction Chains. *Inform. Theor. Appl.*, 24:531–543, 1990.
11. J. A. Solinas. An Improved Algorithm for Arithmetic on a Family of Elliptic Curves. In Burton S. Kaliski Jr., editor, *Advances in Cryptology - Proceedings of CRYPTO 1997*, volume 1294 of *Lecture Notes in Computer Science*, pages 357–371. Springer, 1997.
12. R. Tijdeman. On the maximal distance between integers composed of small primes. *Comp. Mathematica*, 28:159–162, 1974.