Improved Algorithms for Arithmetic on Anomalous Binary Curves*

Jerome A. Solinas

National Security Agency, Ft. Meade, MD 20755, USA
Visitor, Centre for Applied Cryptographic Research

Abstract. It has become increasingly common to implement discrete-logarithm based public-key protocols on elliptic curves over finite fields. The basic operation is *scalar multiplication:* taking a given integer multiple of a given point on the curve. The cost of the protocols depends on that of the elliptic scalar multiplication operation.

Koblitz introduced a family of curves which admit especially fast elliptic scalar multiplication. His algorithm was later modified by Meier and Staffelbach. We give an improved version of the algorithm which runs 50% faster than any previous version. It is based on a new kind of representation of an integer, analogous to certain kinds of binary expansions. We also outline further speedups using precomputation and storage.

Keywords: elliptic curves, exponentiation, public-key cryptography.

1 Introduction

It has become increasingly common to implement discrete-logarithm based public-key protocols on elliptic curves over finite fields. More precisely, one works with the points on the curve, which can be added and subtracted. If we add the point P to itself n times, we denote the result by nP. The operation of computing nP from P is called *scalar multiplication* by n. Elliptic public-key protocols are based on scalar multiplication, and the cost of executing such protocols depends mostly on the complexity of the scalar multiplication operation.

Scalar multiplication on an elliptic curve is analogous to exponentiation in the multiplicative group of integers modulo a fixed integer m. Various techniques have been developed [Gor98] to speed modular

* This paper is a corrected and updated version of the paper appearing in the Proceedings of Crypto '97.

exponentiation using memory and precomputations. Such methods, for the most part, carry over to elliptic scalar multiplication.

There are also efficiency improvements available in the elliptic case that have no analogue in modular exponentiation. There are three kinds of these:

1. One can choose the curve, and the base field over which it is defined, so as to optimize the efficiency of elliptic scalar multiplication. Thus, for example, one might choose the field of integers modulo a Mersenne prime, since modular reduction is particularly efficient [Knu81] in that case. This option is not available for, say, RSA systems, since the secret primes are chosen randomly in order to maintain the security of the system.
2. One can use the fact that subtraction of points on an elliptic curve is just as efficient as addition. (The analogous statement for integers (mod m) is false, since modular division is more expensive than modular multiplication.) The efficient methods for modular exponentiation all involve a sequence of squarings and multiplications that is based on the binary expansion of the exponent. The analogous procedure for elliptic scalar multiplication uses a sequence of doublings and additions of points. If we allow subtractions of points as well, we can replace [MO90] the binary expansion of the coefficient n by a more efficient *signed binary expansion* (*i.e.* an expansion in powers of two with coefficients 0 and ± 1).
3. One can use *complex multiplication.* Every elliptic curve over a finite field[1] comes equipped with a set of operations which can be viewed as multiplication by complex algebraic integers (as opposed to ordinary integers). These operations can be carried out efficiently for certain families of elliptic curves. In these cases, they can be utilized in various ways [Kob91] to increase the efficiency of elliptic scalar multiplication.

It is the purpose of this paper to present a new technique for elliptic scalar multiplication. This new algorithm incorporates elements from all three of the above categories. The new method is

[1] We restrict our attention to elliptic curves that are not *supersingular*, since such curves are cryptographically weaker than ordinary curves [MOV91]. But see [Kob98] for cryptographic applications of supersingular curves.

50% faster than any method previously known for operating on a non-supersingular elliptic curve.

2 Field and Elliptic Operations in $\mathbb{F}_{2^m}$

We begin with a brief survey of the various operations we will need in the field $\mathbb{F}_{2^m}$ and on elliptic curves over this field.

Squaring. We will assume that the field $\mathbb{F}_{2^m}$ is represented in terms of a *normal basis:* a basis over $\mathbb{F}_2$ of the form

$$\left\{\theta, \theta^2, \theta^{2^2}, \ldots, \theta^{2^{m-1}}\right\} \ .$$

The advantage of this representation is that squaring a field element can be accomplished by a one-bit cyclic shift of the bit string representing the element. This property will be crucial in what follows. If m is not divisible by 8, then one can use Gaussian cyclotomic periods to construct easily [ABV89] an efficient normal basis for $\mathbb{F}_{2^m}$. (Since our application will require m to be prime, we can always use the Gaussian method.)

Our emphasis in this paper will be the case in which the field arithmetic is implemented in hardware. Although the algorithms that follow will be efficient in software as well, the full advantage of our method occurs in hardware, where the bit shifts (and therefore field squarings) are virtually free.

Addition and Multiplication. We may neglect the cost of additions in $\mathbb{F}_{2^m}$ since they involve only bitwise XORs. A multiplication (of distinct elements) takes about m times as long, just as in the case of integer arithmetic. The cost of an elliptic operation depends mostly on the number of field multiplications it uses.

Inversion. Multiplicative inversion in $\mathbb{F}_{2^m}$ can be performed in

$$L(m-1) + W(m-1) - 2$$

field multiplications using the method of [ITT86]. Here $L(k)$ represents the length of the binary expansion of k, and $W(k)$ the number

of ones in the expansion. This fact may be a consideration when choosing the degree m. (Alternatively, one can use the Euclidean algorithm [Ber84], but one must first convert from the normal basis representation to the more familiar polynomial basis form, and then back again after the inversion.)

Elliptic Addition. The standard equation for an elliptic curve over $\mathbb{F}_{2^m}$ is the *Weierstrass equation*

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

where $b \neq 0$. Public key protocols based on this curve work on the group consisting of the points (x, y) on this curve, along with the group identity $\mathcal{O}$. (The element $\mathcal{O}$ is called the *point at infinity*, but it is most convenient to represent it[2] by $(0, 0)$.)

Routine 2 *(Elliptic Group Operation)*

Input:
Points P_0 and P_1 on E

Output:
The sum $P_2 := P_0 + P_1$

Computation:

```
If P_0 = O then output P_2 ← P_1 and stop
If P_1 = O then output P_2 ← P_0 and stop
If x_0 = x_1
     then
          if y_0 + y_1 = x_1
               then
                    output O and stop
               else
                    set λ ← x_1 + y_1/x_1
                        x_2 ← λ^2 + λ + a
                        y_2 ← x_1^2 + (λ + 1) x_2
     else
          set λ ← (y_0 + y_1)/(x_0 + x_1)
```

[2] This does not cause confusion, because the origin is never on E.

$$x_2 \leftarrow \lambda^2 + \lambda + x_0 + x_1 + a$$
$$y_2 \leftarrow (x_1 + x_2)\,\lambda + x_2 + y_1$$
`Output` $P_2 \leftarrow (x_2, y_2)$

To *subtract* the point $P = (x, y)$, one adds the point $-P = (x, x+y)$.

Except for the special cases involving $\mathcal{O}$, the above addition and subtraction operations each require 1 multiplicative inversion and 2 multiplications.[3] (As always, we disregard the cost of adding and squaring field elements.)

3 Elliptic Scalar Multiplication

We next discuss the common methods for performing scalar multiplication on an arbitrary elliptic curve. These results will not be necessary for the subject of this paper, but will serve to motivate the new algorithms, which are analogues of these methods.

3.1 The Addition-Subtraction Method

The basic technique for elliptic scalar multiplication is the *addition-subtraction method.* It is based on the *nonadjacent form* (NAF) of the coefficient n: a signed binary expansion with the property that no two consecutive coefficients are nonzero. For example,

$$\mathrm{NAF}(29) = \langle 1, 0, 0, -1, 0, 1 \rangle \tag{3}$$

since $29 = 32 - 4 + 1$.

Just as every positive integer has a unique binary expansion, it also has a unique NAF. Moreover, $\mathrm{NAF}(n)$ has the fewest nonzero coefficients of any signed binary expansion of n (see [Gor98]). There are several ways to construct the NAF of n from its binary expansion. We present the one that most resembles the new algorithm we will present in §4.

The idea is to divide repeatedly by 2. Recall that one can derive the binary expansion of an integer by dividing by 2, storing off the remainder (0 or 1), and repeating the process with the quotient. To derive a NAF, one allows remainders of 0 or ± 1. If the remainder is to be ± 1, one chooses whichever makes the quotient even.

[3] There do exist special-purpose improvements to the basic elliptic operations (see *e.g.* [Lop99]), but they are not relevant to this paper.

Routine 4 *(NAF)*

Input:
a *positive integer* n

Output:
NAF(n)

Computation:

```
Set c ← n
Set S ← ⟨ ⟩
While c > 0
    If c odd
        then
            set u ← 2 − (c mod 4)
            set c ← c − u
        else
            set u ← 0
    Prepend u to S
    Set c ← c/2
EndWhile
Output S
```

For example, to derive (3), one applies (4) with $n = 29$. The results are shown in Fig. 1.

Note that, although we have phrased the algorithm in terms of integer arithmetic, it can be implemented in terms of bit operations on the binary expansion of n. No arithmetic operations are needed beyond integer addition by 1.

In the derivation of the ordinary binary expansion, the sequence c is decreasing, but that is not true in general in (4). As a result, the NAF of a number may be longer than its binary expansion. Fortunately, it can be at most one bit longer, because

$$2^\ell < 3n < 2^{\ell+1} \tag{5}$$

where ℓ is the bit length of NAF(n). (See [MO90].)

The routine (4) can be modified as follows to produce an algorithm for elliptic scalar multiplication.

Fig. 1. Computing a NAF.

c	u	$\mathcal{S}$
29		$\langle\,\rangle$
28	1	
14		$\langle 1\rangle$
14	0	
7		$\langle 0,1\rangle$
8	-1	
4		$\langle -1,0,1\rangle$
4	0	
2		$\langle 0,-1,0,1\rangle$
2	0	
1		$\langle 0,0,-1,0,1\rangle$
0	1	
0		$\langle 1,0,0,-1,0,1\rangle$

Routine 6 *Addition-Subtraction Method*

Input:

a positive integer n
an elliptic curve point P

Output:

the point nP

Computation:

```
Set c ← n
Set Q ← O,   P_0 ← P
While c > 0
    If c odd then
        set u ← 2 − (c mod 4)
        set c ← c − u
        if u = 1 then set Q ← Q + P_0
        if u = −1 then set Q ← Q − P_0
    Set c ← c/2
    Set P_0 ← 2P_0
EndWhile
Output Q
```

This algorithm is a *right-to-left* method, since (4) builds up the NAF starting at the least significant bit and ending at the most significant.[4] It is possible to give a *left-to-right* addition-subtraction method, but it has the disadvantage that it requires the entire NAF to be computed first, thus requiring more storage space and memory calls.

The cost of the addition-subtraction method depends on the bit length ℓ of NAF(n), which we now estimate. It follows from the Hasse theorem [Sil86] that the order of an elliptic curve over $\mathbb{F}_{2^m}$ is

$$\#E(\mathbb{F}_{2^m}) = 2^m + O(2^{m/2}) \ .$$

Most public-key protocols on elliptic curves use a base point of prime order r. Since all of the curves (1) have even order, then

$$r \leq 2^{m-1} + O(2^{m/2}) \ .$$

We can assume that $n < r$; indeed, by using the identity

$$n(x, y) = (r - n)(x, x + y) \ ,$$

we can assume that $n < r/2$. Thus $\ell < m$, so that (6) requires about m doubles at most.

The number of additions is one less than the Hamming weight of (*i.e.* number of nonzero coefficients in) NAF(n). The average density of nonzero coefficients among NAF's of length ℓ is

$$\frac{2^\ell\,(3\,\ell - 4) - (-1)^\ell\,(6\,\ell - 4)}{9\,(\ell - 1)\,(2^\ell - (-1)^\ell)} \ , \tag{7}$$

or approximately (and asymptotically) $1/3$ (see [MO90]). It follows via (5) that the Hamming weight H of NAF(n) satisfies

$$H \approx \frac{1}{3}\,\log_2 n \ . \tag{8}$$

Therefore, the average cost of (6) is $\sim m$ doubles and $\sim m/3$ additions, for a total of $\sim 4m/3$ elliptic operations. This compares

[4] It is easy to prove there is no left-to-right method for computing the NAF. On the other hand, there exist signed binary expansions that are as good as the NAF and that can be computed from left to right.

favorably to the classical *binary method*, which uses the ordinary binary expansion in place of the NAF. For binary expansions, the average density is exactly 1/2 rather than the value (7); thus the binary method requires about 12% more elliptic operations than the addition-subtraction method.

3.2 Window Methods

The addition-subtraction method can be generalized to produce still more efficient algorithms provided extra memory is available and precomputation is permitted. We present the basic method, called the *width-w window method.*[5]

Let w be an integer greater than 1. Then each positive integer has a unique *width-w NAF:* an expression

$$n = \sum_{j=0}^{\ell-1} u_j \, 2^j$$

where:

- each nonzero u_j is odd and less than 2^{w-1} in absolute value;
- among any w consecutive coefficients, at most one is nonzero.

The case $w = 2$ is that of the ordinary NAF. The width-w NAF is written

$$\mathrm{NAF}_w(n) = \langle u_{\ell-1}, \ldots u_0 \rangle \ .$$

It can be computed via the following generalization of (4).

Routine 9 *(Width-w NAF)*

Input:
a *positive integer* n

Output:
$\mathrm{NAF}_w(n)$

[5] More elaborate window methods exist (see [Gor98]), but they can require a great deal of initial calculation and seldom do much better than the technique presented here.

Computation:

```
Set c ← n
Set S ← ⟨ ⟩
While c > 0
    If c odd
        then  set u ← c mods 2^w
              set c ← c − u
        else  set u ← 0
    Prepend u to S
    Set c ← c/2
EndWhile
Output S
```

The notation

$$a := b \mod c$$

means that a is the integer satisfying

$$a \equiv b \pmod{c}$$

and

$$-\frac{c}{2} \leq a < \frac{c}{2} \ .$$

The routine (9) is commonly described by saying that one slides a "window" of width w along the binary expansion from right to left, using the contents to output the next entry of $\mathrm{NAF}_w(n)$.

Given the width-w NAF, one can perform elliptic scalar multiplication by n via the following algorithm.

Routine 10 *Width-w Addition-Subtraction Method*

Input:
a *positive integer* n
an *elliptic curve point* P

Output:
the point nP

Precomputation:

```
Set P_0 ← P
Set P_{2^{w-2}-1} ← 2 P_0
For i from 1 to 2^{w-2} − 1 do
    Set P_i ← P_{i-1} + P_{2^{w-2}-1}
Next i
```

Computation:

```
Compute NAF_w(n) = ⟨u_{ℓ-1}, ..., u_0⟩      (via (9))
Set Q ← O
For j from ℓ − 1 downto 0 do
    Set Q ← 2Q
    If u_j ≠ 0 then
        i ← (|u_j| − 1)/2
        if u_j > 0
            then set Q ← Q + P_i
            else set Q ← Q − P_i
Output Q
```

The routine (10) is a left-to-right algorithm. The right-to-left algorithm (6) does not generalize well to the width-w case, since each point P_i would have to be doubled ℓ times. As remarked in [Mül98], this is a general difficulty with window methods: the binary expansions must be computed right to left in general, but the elliptic scalar multiplication is best done from left to right. This difficulty will not arise with the particular curves that are the subject of this paper.

The average density of a width-w NAF is $(w+1)^{-1}$. Thus one can diminish greatly the number of elliptic additions in an elliptic scalar multiplication, provided the memory is available. These speedups, however, do nothing to reduce the number of elliptic doublings. There seems to be little that can be done about this in general. The following special curves, however, admit elliptic scalar multiplication that do not use doublings at all.

4 Anomalous Binary Curves

The *anomalous binary curves* (or ABC's) are the curves E_0 and E_1 defined over $\mathbb{F}_2$ by

$$E_a : y^2 + xy = x^3 + a\,x^2 + 1 \ .$$

They are also called *Koblitz curves* since their efficient scalar multiplication properties were first presented in [Kob91].

4.1 Basic Properties

We survey the basic properties of ABC's that we will need.

Group Orders. We denote by $E_a(\mathbb{F}_{2^m})$ the group of $\mathbb{F}_{2^m}$-rational points on E_a. This is the group on which the public-key protocols are performed. The group should be chosen so that it is computationally difficult to compute discrete logarithms of its elements. Thus, for example, the order $\#E_a(\mathbb{F}_{2^m})$ should be divisible by a large prime (see [MOV97]). Ideally, $\#E_a(\mathbb{F}_{2^m})$ should be a prime or the product of a prime and small integer. This can only happen when m is itself prime, for otherwise there are large divisors arising from subgroups $E_a(\mathbb{F}_{2^d})$ where d divides m.

When m is prime, the only such divisor is that arising from $d = 1$. The ABC's over $\mathbb{F}_2$ are

$$\begin{aligned} E_1(\mathbb{F}_2) &= \{\mathcal{O}, (0,1)\} \\ E_0(\mathbb{F}_2) &= \{\mathcal{O}, (0,1), (1,0), (1,1)\} \ . \end{aligned}$$

Since $E_a(\mathbb{F}_2)$ is a subgroup of $E_a(\mathbb{F}_{2^m})$, it follows that the order $\#E_a(\mathbb{F}_{2^m})$ is always divisible by

$$f = \#E_a(\mathbb{F}_2) = \begin{cases} 2 & \text{for } a = 1 \\ 4 & \text{for } a = 0 \ . \end{cases} \tag{11}$$

We define an integer to be *very nearly prime* if it is of the form $N = f \cdot r$, where $f = 2$ or 4 and $r > 2$ is prime. Although the orders $\#E_a(\mathbb{F}_{2^m})$ are never prime for $m > 1$, they are frequently very nearly

prime. The values of $m \leq 512$ for which $\#E_1(\mathbb{F}_{2^m})$ is twice a prime are

$$m = 3, 5, 7, 11, 17, 19, 23, 101, 107, 109,$$
$$113, 163, 283, 311, 331, 347, 359 \ .$$

The values of $m \leq 512$ for which $\#E_0(\mathbb{F}_{2^m})$ is 4 times a prime are

$$m = 5, 7, 13, 19, 23, 41, 83, 97, 103, 107,$$
$$131, 233, 239, 277, 283, 349, 409 \ .$$

(The calculation of the orders is quite simple; the technique is given below.) The curves with very nearly prime order are the ones of most cryptographic interest.

The Main Subgroup. Suppose that $\#E_a(\mathbb{F}_{2^m}) = f \cdot r$ is very nearly prime. We define the *main subgroup* to be the subgroup of order r. It is common (see *e.g.* [IEEE99]) to perform cryptographic operations in the main subgroup rather than the entire curve.

Proposition 12 *Suppose that $\#E_a(\mathbb{F}_{2^m})$ is very nearly prime, and let P be a point on $E_a(\mathbb{F}_{2^m})$. Then P is in the main subgroup if and only if $P = fQ$ for some Q on $E_a(\mathbb{F}_{2^m})$.*

Proof. Both curves $E_a(\mathbb{F}_2)$ are cyclic groups; to see this, one need only observe that $2\,(1,0) = (1,1)$ on $E_1(\mathbb{F}_2)$. Thus the curve $E_a(\mathbb{F}_{2^m})$ is cyclic whenever its order is very nearly prime. The result follows from the standard properties of finite cyclic groups. □

As a result of (12), we have the following simple conditions to determine whether a given point is in the main subgroup. If $a = 1$, then a point $P = (x, y)$ is in the main subgroup if and only if $\mathrm{Tr}(x) = 1$ (see [Ser98]). If $a = 0$, then (x, y) is in the main subgroup if and only if $\mathrm{Tr}(x) = 0$ and $\mathrm{Tr}(y) = \mathrm{Tr}(\lambda\, x)$, where λ is an element with $\lambda^2 + \lambda = x$. (See Appendix A for proofs.) With a normal basis representation of the field, both the trace and the computation of λ can be done very efficiently. Thus, checking for membership in the main subgroup is essentially free if $a = 1$, and costs only one field multiplication if $a = 0$.

Because the main subgroup is the object of cryptographic interest, it is most important to optimize the elliptic scalar multiplication operation there. In §7.2, an algorithm for elliptic scalar multiplication in the main subgroup will be presented.

Complex Multiplication. Since the ABC's are defined over $\mathbb{F}_2$, they have the following property: *if $P = (x, y)$ is a point on E_a, then so is the point (x^2, y^2)*. Moreover, one can verify from (2) that

$$(x^4, y^4) + 2\,(x, y) = \mu \cdot (x^2, y^2) \tag{13}$$

for every (x, y) on E_a, where

$$\mu := (-1)^{1-a} \ . \tag{14}$$

This relation can be written more easily in terms of the Frobenius (squaring) map over $\mathbb{F}_2$:

$$\tau(x, y) := (x^2, y^2) \ . \tag{15}$$

Using this notation, (13) becomes

$$\tau\,(\tau\,P) + 2\,P = \mu\,\tau\,P$$

for all $P \in E_a$. Symbolically, this can be written

$$(\tau^2 + 2)\,P = \mu\,\tau\,P \ .$$

This means that the squaring map can be regarded as implementing multiplication by the complex number τ satisfying

$$\tau^2 + 2 = \mu\,\tau \ .$$

Explicitly, this number is

$$\tau = \frac{\mu + \sqrt{-7}}{2} \ .$$

By combining the squaring map with ordinary scalar multiplication, we can multiply points on E_a by any element of the ring $\mathbb{Z}[\tau]$. We say that E_a has *complex multiplication* by τ. (See [Kob91].)

Lucas Sequences. The *Lucas sequences* are sequences of integers that facilitate computations involving quadratic irrationals. The Lucas sequences for the numbers τ will be used frequently in what follows; thus we summarize here their relevant properties.

- There are two Lucas sequences, U_k and V_k, associated with a given quadratic irrational. For τ, the sequences are defined as follows.

$$\begin{aligned} U_0 = 0, \quad U_1 = 1 \quad \text{and} \quad U_{k+1} = \mu\, U_k - 2\, U_{k-1} \quad \text{for } k \geq 1; \\ V_0 = 2, \quad V_1 = \mu \quad \text{and} \quad V_{k+1} = \mu\, V_k - 2\, V_{k-1} \quad \text{for } k \geq 1. \end{aligned} \tag{16}$$

- It can be proved by induction that

$$\begin{aligned} U_k &= \left(\tau^k - \overline{\tau}^k\right) / \sqrt{-7} \\ V_k &= \tau^k + \overline{\tau}^k \ . \end{aligned} \tag{17}$$

If $\tan\theta = \sqrt{7}$, then (17) can be written as

$$\begin{aligned} U_k &= \mu^{k+1}\, 2^{k/2+1}\, \sin(k\,\theta)/\sqrt{7} \\ V_k &= \mu^k\, 2^{k/2+1}\, \cos(k\,\theta) \ . \end{aligned} \tag{18}$$

- It can be proved by induction that

$$\tau^k = U_k\, \tau - 2\, U_{k-1} \text{ for } k \geq 1 \ . \tag{19}$$

Multiplying this equation by its conjugate, one obtains

$$U_k^2 - \mu\, U_k\, U_{k-1} + 2\, U_{k-1}^2 = 2^{k-1} \text{ for } k \geq 1 \ . \tag{20}$$

- The group orders $\#E_a(\mathbb{F}_{2^m})$ are easily computed via

$$\#E_a(\mathbb{F}_{2^m}) = 2^m + 1 - V_m \ . \tag{21}$$

This identity follows from the basic properties of zeta functions of curves; see *e.g.* [Kob94].

The Norm. The *norm* of an element $\alpha \in \mathbb{Z}[\tau]$ is the product of α and its complex conjugate $\overline{\alpha}$. Explicitly, the norm of $\delta := d_0 + d_1\,\tau$ is

$$N(\delta) = d_0^2 + \mu\, d_0\, d_1 + 2\, d_1^2 \ .$$

We will require the following properties of the norm.

- The norm function satisfies

$$N(\alpha\,\beta) = N(\alpha)\,N(\beta) \tag{22}$$

for all α, β in $\mathbb{Z}[\tau]$.

- The Euclidean distance from α to 0 in the complex plane is given by $\sqrt{N(\alpha)}$. Thus the Triangle Inequality takes the form

$$\sqrt{N(\alpha+\beta)} \le \sqrt{N(\alpha)} + \sqrt{N(\beta)}\ . \tag{23}$$

- We will require the norms of some specific elements. It is easily checked that

$$N(\tau) = 2$$

and that

$$N(\tau - 1) = f$$

where f is as given in (11).

Finally,

$$N(\tau^m - 1) = \#E_a(\mathbb{F}_{2^m}) \tag{24}$$

(see [Sil86]). (As an alternative to (21), the order of an ABC can be computed using this identity and (19).)

Since $\tau - 1$ divides $\tau^m - 1$, it follows that f divides $\#E_a(\mathbb{F}_{2^m})$ and that

$$N\left((\tau^m - 1)/(\tau - 1)\right) = \#E_a(\mathbb{F}_{2^m})/f\ . \tag{25}$$

- The ring $\mathbb{Z}[\tau]$ is Euclidean with respect to the norm function (see [ST87]). That is, given an element γ and a nonzero element δ, there exist elements κ and ρ such that $\gamma = \delta\,\kappa + \rho$ and

$$N(\rho) < N(\delta)\ . \tag{26}$$

As a result of this property, the ring $\mathbb{Z}[\tau]$ has unique factorization. The element τ, having prime norm, is a prime element.

4.2 The τ-adic NAF

The complex multiplication property is useful for elliptic scalar multiplication because multiplication by τ, being implemented by squaring, is essentially free when $\mathbb{F}_{2^m}$ is represented in terms of a normal basis.[6] Thus it is worthwhile, when computing nP, to regard n as

[6] As R. Schroeppel has remarked, these algorithms are also useful when using a polynomial basis, since squaring is still relatively efficient in that case.

an element of $\mathbb{Z}[\tau]$ rather than as "just" an integer. More precisely, one replaces the (signed) binary expansion of the coefficient with a (signed) *τ-adic expansion.* That is, one represents n as a sum and difference of distinct powers of τ.

For example, with $a = 1$ we have

$$9 = \tau^5 - \tau^3 + 1 \ . \tag{27}$$

Thus, if $P = (x, y)$ is a point on E_1, then

$$9P = (x^{32}, y^{32}) - (x^8, y^8) + (x, y) \ .$$

The above example gives 9 as what we call a *τ-adic NAF*, since no two consecutive terms are nonzero. (Both [Kob91] and [MS93] use signed τ-adic expansions, but neither kind has the nonadjacency property.) As we shall see, the use of τ-adic NAF's gives a significant reduction in the number of terms, just as NAF's give a significant improvement over binary expansions in the case of integers.

The remainder of this section is dedicated to proving the following result, which is analogous to the case of the ordinary NAF for integers.

Theorem 1. *Every element of the ring $\mathbb{Z}[\tau]$ has a unique τ-adic NAF.*

We can therefore speak of *the* τ-adic NAF of an element α. We will denote it by $\mathrm{TNAF}(\alpha)$. Thus (27) is written

$$\mathrm{TNAF}(9) = \langle 1, 0, -1, 0, 0, 1 \rangle \ .$$

In the course of proving Thm. 1, we will develop an efficient algorithm for computing the τ-adic NAF for any element of $\mathbb{Z}[\tau]$.

Lemma 28 *The element $c_0 + c_1 \tau$ of $\mathbb{Z}[\tau]$ is divisible by τ if and only if c_0 is even. It is divisible by τ^2 if and only if*

$$c_0 \equiv 2\, c_1 \pmod 4 \ . \tag{29}$$

Proof. If μ is defined as in (14), then

$$(d_0 + d_1\,\tau)\,\tau = -2\,d_1 + (d_0 + \mu\,d_1)\,\tau \ .$$

It follows at once that, if τ divides $c_0 + c_1\,\tau$, then c_0 is even.

Conversely, if c_0 is even, then

$$\frac{c_0 + c_1\,\tau}{\tau} = \frac{\mu\,c_0 + 2\,c_1}{2} - \frac{c_0}{2}\,\tau$$

is an element of $\mathbb{Z}[\tau]$.

To prove the corresponding statements for τ^2, we begin with the identity

$$\tau^2 = \mu\,\tau - 2 \ .$$

It follows that every multiple of τ^2 has the form

$$(d_0 + d_1\,\tau)\,(\mu\,\tau - 2) = -2\,(d_0 + \mu\,d_1) + (\mu\,d_0 - d_1)\,\tau \ .$$

It is easily verified that the values

$$\begin{aligned} c_0 &= -2\,(d_0 + \mu\,d_1) \\ c_1 &= \mu\,d_0 - d_1 \end{aligned}$$

satisfy (29). Conversely,

$$\frac{c_0 + c_1\,\tau}{\tau^2} = -\frac{(1 + 2\,\mu)\,c_0 + 2\,\mu\,c_1}{4} + \mu \cdot \frac{c_0 - 2\,c_1}{4}\,\tau \ ,$$

which is easily seen to be an element of $\mathbb{Z}[\tau]$ if (29) holds. □

In light of (28), we have the following algorithm (joint work with R. Reiter) for computing the τ-adic NAF. It is completely analogous to (4), but here we are dividing by τ rather than by 2. Since τ has norm 2, then by (26), the possible remainders upon division by τ are ± 1. Earlier algorithms chose the remainder that minimized the norm of the quotient; this is analogous to the basic division algorithm for generating the binary expansion of an integer. What we shall do instead is to choose the remainder that makes the quotient divisible by τ. This is analogous to the computation of the NAF for integers.

Algorithm 1 *(τ-adic NAF)*

Input:
integers r_0, r_1

Output:
$\mathrm{TNAF}(r_0 + r_1\,\tau)$

Computation:

```
Set c_0 ← r_0 , c_1 ← r_1
Set S ← ⟨ ⟩
While c_0 ≠ 0 or c_1 ≠ 0
    If c_0 odd
        then
            set u ← 2 − (c_0 − 2 c_1 mod 4)
            set c_0 ← c_0 − u
        else
            set u ← 0
    Prepend u to S
    Set (c_0, c_1) ← (c_1 + μ c_0/2, −c_0/2)
EndWhile
Output S
```

For example, to derive (27), one applies Alg. 1 with $a = 1$, $c_0 = 9$, and $c_1 = 0$. The results are shown in Fig. 2.

Note that the implementation of Alg. 1 involves nothing more complicated than integer addition. (This is slightly more than is required by (4), which only adds 1 to an integer.)

Having described how to compute a τ-adic NAF, we now prove that such a representation is unique.

Lemma 30 *Let $\alpha \in \mathbb{Z}[\tau]$. Then precisely one of the following statements hold:*

- *α is divisible by τ.*
- *$\alpha \equiv 1 \pmod{\tau^2}$.*
- *$\alpha \equiv -1 \pmod{\tau^2}$.*

Fig. 2. Computing a τ-adic NAF.

c_0	c_1	u	$\mathcal{S}$
9	0		$\langle\,\rangle$
8	0	1	
4	−4		$\langle 1\rangle$
4	−4	0	
−2	−2		$\langle 0,1\rangle$
−2	−2	0	
−3	1		$\langle 0,0,1\rangle$
−2	1	−1	
0	1		$\langle -1,0,0,1\rangle$
0	1	0	
1	0		$\langle 0,-1,0,0,1\rangle$
0	0	1	
0	0		$\langle 1,0,-1,0,0,1\rangle$

Proof. Let $\alpha = c_0 + c_1\tau$. If c_0 is even, then τ divides α by (28). If c_0 is odd, then precisely one of $c_0 \pm 1$ satisfies (29); thus τ^2 divides precisely one of $\alpha \pm 1$, by (28). □

Corollary 31 *Let $\alpha \in \mathbb{Z}[\tau]$. Then any two τ-adic NAF's of α have the same rightmost entry.*

Proof. Depending on which of the three possibilities hold for α, the rightmost entry of a τ-adic NAF for α is 0, 1, or -1, respectively. □

Proof of Thm. 1. The existence of the τ-adic NAF is established by Alg. 1. The uniqueness follows from (31), using induction on the length of the τ-adic NAF. □

4.3 Length and Density of τ-adic NAF's

To evaluate the usefulness of the τ-adic NAF as a substitute for an ordinary NAF, it is necessary to know its Hamming weight (*i.e.* number of nonzero terms). In the case of the ordinary NAF, the weight is calculated in terms of the length of the NAF (given by (5)) and the density of nonzero terms (given by (7)). We now obtain analogous results for τ-adic NAF's.

The calculation of the density is trivial.

Proposition 32 *The average density among τ-adic NAF's of length ℓ is given by (7), and is therefore asymptotically* $1/3$.

Proof. The result follows from (7) since the same sequences occur as length-ℓ NAF's and length-ℓ τ-adic NAF's. □

The estimation of the length of a NAF is more involved. Intuitively, the answer should be

$$\ell \approx \log_2(N(\alpha)) \ ,$$

because Alg. 1 begins with α, and divides by the norm-2 element τ in computing each entry of the τ-adic NAF. Of course this ignores the effect of the additions and subtractions, but provides a benchmark for the results to follow.

We begin with some notation and terminology. By the *length* of an element of $\mathbb{Z}[\tau]$ we mean the length of its τ-adic NAF. Let $N_{\max}(k)$ denote the largest norm occurring among all length-k elements of $\mathbb{Z}[\tau]$.

Lemma 33 *For all k,*

$$2\, N_{\max}(k) \leq N_{\max}(k+1) \ .$$

Proof. If α is a length-k element of maximal norm, then $\tau\alpha$ is an element of length $k+1$ and norm $2\, N(\alpha)$. □

Corollary 34 *$N_{\max}(k)$ is the largest norm occurring among all elements α of $\mathbb{Z}[\tau]$ whose length is at most k.*

Lemma 35 *If $c > e$, then*

$$\sqrt{N_{\max}(c)} \leq 2^{e/2} \sqrt{N_{\max}(c-e)} + \sqrt{N_{\max}(e)} \ .$$

Proof. Let γ be a length-c element of maximal norm, so that

$$N(\gamma) = N_{\max}(c) \ . \tag{36}$$

Let the τ-adic NAF of γ be

$$\mathrm{TNAF}(\gamma) = \langle u_{c-1}, \ldots, u_0 \rangle \ , \tag{37}$$

and define ρ by

$$\mathrm{TNAF}(\rho) = \langle u_{e-1}, \ldots, u_0 \rangle \ . \tag{38}$$

Then ρ has length at most e, so that

$$N(\rho) \leq N_{\max}(e) \tag{39}$$

by (34). Finally,

$$\gamma = \tau^e \, \delta + \rho \tag{40}$$

for some δ of length $c - e$; note that

$$N(\delta) \leq N_{\max}(c - e) \ . \tag{41}$$

It follows from the Triangle Inequality (23) that

$$\sqrt{N(\gamma)} \leq 2^{e/2} \sqrt{N(\delta)} + \sqrt{N(\rho)} \ .$$

The result now follows by (36), (39), and (41). □

For convenience we introduce the notation

$$M_k := \sqrt{2^{-k} \, N_{\max}(k)} \ .$$

In this notation, (33) states that

$$M_k \leq M_{k+1} \tag{42}$$

for all k, and (35) states that

$$M_c \leq M_{c-e} + 2^{-(c-e)/2} \, M_e \tag{43}$$

if $c > e$.

Proposition 44 *For d and q positive,*

$$\frac{M_{dq}}{1 - 2^{-dq/2}} \leq \frac{M_d}{1 - 2^{-d/2}} \ .$$

Proof. It follows from (43) that

$$M_{(k+1)\,d} - M_{kd} \leq 2^{-kd/2}\, M_d$$

for $k = 1, \ldots, (q-1)$. Summing over k, we obtain

$$M_{dq} - M_d \leq M_d \sum_{k=1}^{q-1} \left(2^{-d/2}\right)^k \quad .$$

The result follows from summing the geometric series. □

Proposition 45 *For $\ell > d$,*

$$M_\ell < \frac{M_d}{1 - 2^{-d/2}} \quad .$$

Proof. Let $d\,q$ be the smallest multiple of d greater than or equal to ℓ. Then $M_\ell \leq M_{dq}$ by (42), and

$$M_{dq} < \frac{M_d}{1 - 2^{-d/2}}$$

by (44). □

Corollary 46 *For $\ell > d$,*

$$N_{\max}(\ell) < \frac{N_{\max}(d)}{(2^{d/2} - 1)^2} \cdot 2^\ell \quad .$$

The result (46) provides the basic upper bound for the norm of a length-ℓ element. We now derive a lower bound. Let $N_{\min}(k)$ denote the smallest norm occurring among all length-k elements of $\mathbb{Z}[\tau]$.

Lemma 47 *If $c > e$, then*

$$\sqrt{N_{\min}(c)} \geq 2^{e/2} \sqrt{N_{\min}(c-e)} - \sqrt{N_{\max}(e)} \quad .$$

Proof. Let γ be a length-c element of minimal norm, so that

$$N(\gamma) = N_{\min}(c) \ . \tag{48}$$

Let the τ-adic NAF of γ be as in (37). Define ρ as in (38), and δ as in (40). Then

$$N(\delta) \geq N_{\min}(c-e) \tag{49}$$

and

$$N(\rho) \leq N_{\max}(e) \ , \tag{50}$$

the latter following from (34). By the Triangle Inequality (23),

$$\sqrt{N(\gamma)} \geq 2^{e/2} \sqrt{N(\delta)} - \sqrt{N(\rho)} \ .$$

The result now follows by (48), (49), and (50). □

Corollary 51 *For $\ell > 2\,d$,*

$$N_{\min}(\ell) > \left(\sqrt{N_{\min}(d)} - \frac{\sqrt{N_{\max}(d)}}{2^{d/2}-1} \right)^2 \cdot 2^{\ell-d} \ .$$

Proof. Follows from (46) and (47). □

Combining (46) and (51), we obtain the main result of this section.

Theorem 2. *Let $\ell > 2\,d$, and let α be a length-ℓ element of $\mathbb{Z}[\tau]$. Then*

$$\left(\sqrt{N_{\min}(d)} - \frac{\sqrt{N_{\max}(d)}}{2^{d/2}-1} \right)^2 \cdot 2^{\ell-d} < N(\alpha) < \frac{N_{\max}(d)}{(2^{d/2}-1)^2} \cdot 2^{\ell} \ .$$

To apply Thm. 2, we choose a small value of d, and evaluate $N_{\max}(d)$ and $N_{\min}(d)$ by direct evaluation of all length-d elements of $\mathbb{Z}[\tau]$. This is only feasible if d is quite small, but the resulting bounds are quite accurate.

For example, we apply Thm. 2 with $d = 15$. The bounds are

$$N_{\max}(15) = 47324 \quad \text{and} \quad N_{\min}(15) = 2996 \ .$$

It follows that

$$1.399009614 \cdot 2^{\ell-4} < N(\alpha) < .7301517653 \cdot 2^{\ell+1} \ . \qquad (52)$$

In other words, the length of the τ-adic NAF is bounded by

$$\log_2(N(\alpha)) - .5462682713 < \ell < \log_2(N(\alpha)) + 3.51559412 \qquad (53)$$

when $\ell > 30$.

To measure the quality of these bounds, we compare them with the norms and lengths of some specific (infinite) families of elements.

Let $a = 1$ and

$$\beta := \tau^{14} + \tau^{11} - \tau^9 + \tau^7 - \tau^4 + \tau \ .$$

Then $N(\beta) = 2996$, minimal among elements of length 15. Since

$$N(\beta) \approx (1.4628906) \cdot 2^{11} \ ,$$

it follows that, for any $k \geq 0$, the element $\alpha := \beta\,\tau^k$ of length $\ell := k + 15$ satisfies

$$N(\alpha) \approx (1.4628906) \cdot 2^{\ell-4} \ .$$

This exceeds the lower bound given in (52) by only a small amount.

Now let $a = 1$ and

$$\beta := \tau^{11} - \tau^9 + \tau^6 - \tau^4 + \tau \ .$$

Then $N(\beta) = 5842$, maximal among elements of length up to 12. Let

$$\gamma := (\tau^{13} - 1)\,\beta \ .$$

Clearly, the τ-adic NAF of γ is the concatenation of two copies of that of β, separated by a zero. Thus γ has length 25. Its norm is

$$N(\gamma) = 5842 \cdot N(\tau^{13} - 1)$$

by (22). Now

$$N(\tau^{13} - 1) = 2^{13} + 1 - V_{13}$$

by (24) and (21). Computing the Lucas element V_{13} via (16), we find that

$$N(\tau^{13} - 1) = 8374 \ ,$$

so that

$$N(\gamma) \approx (.7289783) \cdot 2^{26} \ .$$

It follows that, for any $k \geq 0$, the element $\alpha := \gamma \tau^k$ of length $\ell := k + 25$ satisfies

$$N(\alpha) \approx (.7289783) \cdot 2^{\ell+1} \ .$$

This is just slightly under the upper bound given in (52).

It follows from (32) that the average Hamming weight among length-ℓ TNAF's is roughly $\ell/3$. If we assume that this average value holds for the subset of TNAF's of rational integers, then it follows from (53) that the Hamming weight H of the τ-adic NAF for the integer n satisfies

$$H \approx \frac{2}{3} \log_2 n \ . \tag{54}$$

This is twice as large as the Hamming weight of an ordinary NAF, because the τ-adic NAF is twice as long. If we replace the ordinary NAF by the τ-adic NAF, then, we will have eliminated the elliptic doublings in our scalar multiplication method, but doubled the number of elliptic additions. This largely mitigates the advantage of the τ-adic method.

Fortunately, this situation can be fixed. The solution is to replace the τ-adic NAF by an equivalent expression, called a *reduced τ-adic NAF*, that is only half as long.

Before presenting this, however, it is necessary to develop the machinery of modular reduction in the ring $\mathbb{Z}[\tau]$.

5 Modular Reduction in $\mathbb{Z}[\tau]$

In this section, we define precisely what is meant by modular reduction in the ring $\mathbb{Z}[\tau]$, and present an efficient method for implementing it.

Our technique is a generalization of the notion of modular reduction in the ring $\mathbb{Z}$ of rational integers. Suppose c and $d > 1$ are

integers. It is desired to reduce c modulo d, *i.e.* find the integer

$$\rho := c \mod d$$

where the "mods" notation is as in §3.2. The integer ρ can be found by integer division: if

$$\texttt{Round}(\lambda) := \lfloor \lambda + 1/2 \rfloor \ ,$$

then

$$\rho = c - \kappa\, d \ ,$$

where

$$\kappa := \texttt{Round}(c/d) \ .$$

A more compact way of describing ρ is in terms of the *fractional part* operation. The fractional part of λ is defined to be

$$(\!(\lambda)\!) := \lambda - \texttt{Round}(\lambda) \ .$$

The modular reduction process can then be described by

$$\rho = d \ \left(\!\!\left(\frac{c}{d}\right)\!\!\right) \ . \tag{55}$$

Since

$$\left(\!\!\left(\frac{c}{d}\right)\!\!\right) < \frac{1}{2} \ ,$$

it follows that

$$N(\rho) < \frac{N(d)}{2} \ .$$

We now generalize these concepts to the ring $\mathbb{Z}[\tau]$.

5.1 Rounding and Fractional Parts in $\mathbb{Z}[\tau]$

We begin by extending to $\mathbb{Z}[\tau]$ the definitions of $\texttt{Round}(\lambda)$ and $(\!(\lambda)\!)$. (The variable λ now denotes the expression $\lambda_0 + \lambda_1\,\tau$, where the λ_i are real numbers.)

We define $\mathcal{U}$ to be the region in the (λ_0, λ_1)-plane given by the inequalities

$$\begin{aligned} -1 &\le 2\,\lambda_0 + \mu\,\lambda_1 < 1 \\ -2 &\le \lambda_0 + 4\,\mu\,\lambda_1 < 2 \\ -2 &\le \lambda_0 - 3\,\mu\,\lambda_1 < 2 \ . \end{aligned} \tag{56}$$

(See Fig. 3 for $a = 1$ and Fig. 4 for $a = 0$.) Copies of $\mathcal{U}$ tile the plane (see Fig. 5 for the case $a = 1$), with each copy having as its center an element of $\mathbb{Z}[\tau]$. Given $\lambda \in \mathbb{Q}(\tau)$, we "round off" by choosing as $\kappa \in \mathbb{Z}[\tau]$ the center of the copy of $\mathcal{U}$ containing λ. We will denote this operation either by

$$(q_0, q_1) = \texttt{Round}(\lambda_0, \lambda_1)$$

or by

$$q_0 + q_1\,\tau = \texttt{Round}(\lambda_0 + \lambda_1\,\tau) \ ,$$

since there is no danger of confusion.

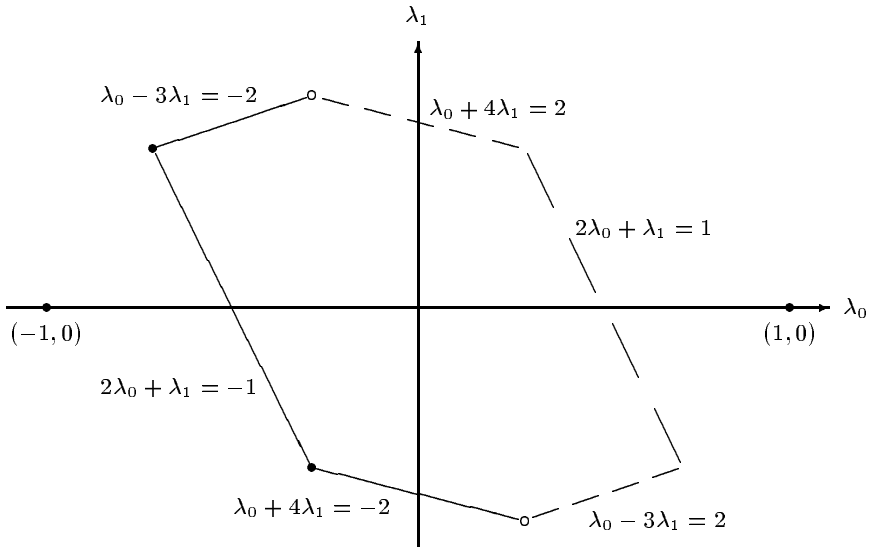

Fig. 3. The region $\mathcal{U}$ for the case $a = 1$.

As in the integer case, we define

$$(\!(\lambda)\!) := \lambda - \texttt{Round}(\lambda)$$

for all complex λ. It is easy to see that the set of possible values for $(\!(\lambda)\!)$ is precisely the region $\mathcal{U}$.

We next prove the main properties of these operations.

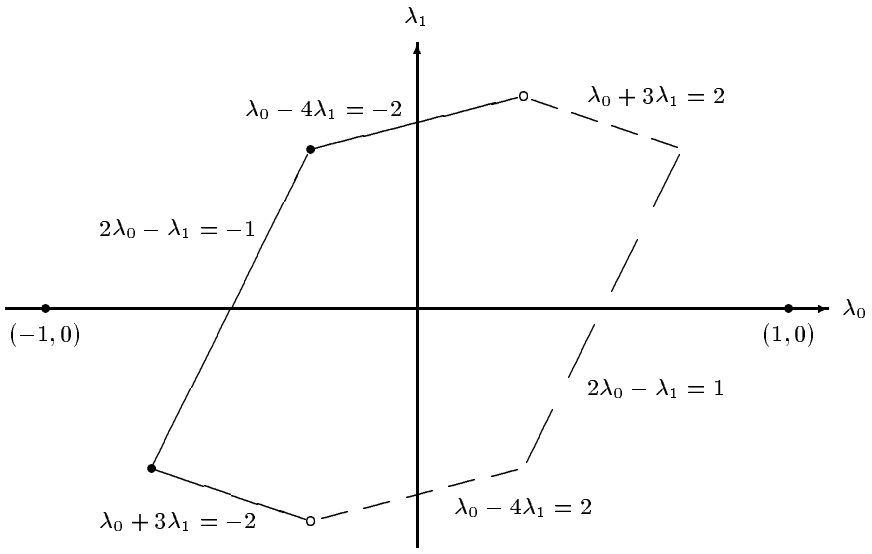

Fig. 4. The region $\mathcal{U}$ for the case $a = 0$.

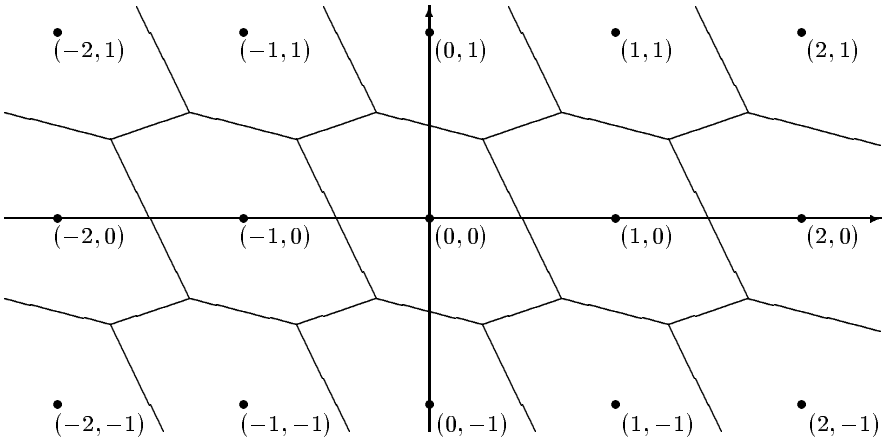

Fig. 5. Copies of $\mathcal{U}$ for the case $a = 1$.

Proposition 57 *Suppose that λ is in the interior of the region $\mathcal{U}$. Then*

$$N(\lambda) < \frac{4}{7} \ .$$

Proof. The set of points in the (λ_0, λ_1)-plane of norm $4/7$ forms the ellipse

$$\lambda_0^2 + \mu\,\lambda_0\,\lambda_1 + 2\,\lambda_1^2 = \frac{4}{7} \ .$$

Now each of the six vertices of $\mathcal{U}$ has norm $4/7$, and so lies on the ellipse. Since $\mathcal{U}$ is convex, it lies entirely in the ellipse. The result follows from the fact that the points inside the ellipse are those of norm less than $4/7$. □

Proposition 58 *Suppose that λ is in the interior of the region $\mathcal{U}$. Then*

$$N(\lambda) < N(\lambda + \alpha)$$

for every nonzero $\alpha \in \mathbb{Z}[\tau]$.

Proof. It is straightforward to check that

$$N(\lambda) < N(\lambda \pm 1) \quad \text{if and only if} \quad |2\,\lambda_0 + \mu\,\lambda_1| < 1 \ ;$$

that

$$N(\lambda) < N(\lambda \pm \tau) \quad \text{if and only if} \quad |\mu\,\lambda_0 + 4\,\lambda_1| < 2 \ ;$$

and that

$$N(\lambda) < N(\lambda \pm \overline{\tau}) \quad \text{if and only if} \quad |\mu\,\lambda_0 - 3\,\lambda_1| < 2 \ .$$

Since λ lies in the interior of $\mathcal{U}$, then by (56) it satisfies all three conditions . Thus the result is proved for $\alpha = \pm 1, \pm\tau, \pm\overline{\tau}$. Now let α be any other nonzero element of $\mathbb{Z}[\tau]$. Then $N(\alpha) \geq 4$; and $N(\lambda) < 4/7$ by (57). Thus the result in this case follows from (23). □

The following properties follow from (57) and (58).

Corollary 59 *If $\kappa := \mathtt{Round}(\lambda)$ and $\zeta := (\!(\lambda)\!) = \lambda - \kappa$, then:*

- $N(\zeta) \leq N(\zeta + \alpha)$ *for every* $\alpha \in \mathbb{Z}[\tau]$.
- $N(\zeta) \leq \frac{4}{7}$.

The first condition of (59) simply says that $\texttt{Round}(\lambda)$ is the element of $\mathbb{Z}[\tau]$ closest to λ, justifying the terminology. The second condition was proved in [MS93].

We now give an algorithm for computing $\texttt{Round}(\lambda)$.

Routine 60 *(Rounding Off)*

Input:
real numbers λ_0, λ_1 *specifying the complex number* $\lambda := \lambda_0 + \lambda_1 \tau$

Output:
real numbers q_0, q_1 *specifying* $q_0 + q_1 \tau := \texttt{Round}(\lambda)$

```
Set  f_0 ← Round(λ_0)
     f_1 ← Round(λ_1)
Set  η_0 ← λ_0 − f_0
     η_1 ← λ_1 − f_1
Set  h_0 ← 0
     h_1 ← 0
Set  η ← 2 η_0 + μ η_1
If   η ≥ 1
     then
          if  η_0 − 3 μ η_1 < −1
               then set  h_1 ← μ
               else set  h_0 ← 1
     else
          if  η_0 + 4 μ η_1 ≥ 2
               then set  h_1 ← μ
If   η < −1
     then
          if  η_0 − 3 μ η_1 ≥ 1
               then set  h_1 ← −μ
               else set  h_0 ← −1
     else
```

```
        if  η_0 + 4 μ η_1 < −2
            then set  h_1 ← −μ
Set  q_0 ← f_0 + h_0
     q_1 ← f_1 + h_1
Output  q_0 , q_1
```

5.2 Division and Modular Reduction in $\mathbb{Z}[\tau]$

We now use the rounding-off operation to develop algorithms for division and modular reduction in $\mathbb{Z}[\tau]$.

The analogue of integer division is as follows. Given a dividend $\gamma = c_0 + c_1\,\tau$ and a divisor $\delta = d_0 + d_1\,\tau$, we wish to find a quotient $\kappa = q_0 + q_1\,\tau$ and a remainder $\rho = r_0 + r_1\,\tau$, such that

$$\gamma = \kappa\,\delta + \rho$$

and such that ρ is as small (in norm) as possible. To do this, we obtain κ by rounding off γ/δ and then solving for ρ. That is, we let

$$\begin{aligned} \lambda &:= \frac{\gamma}{\delta} \\ &= \frac{\gamma\,\overline{\delta}}{N(\delta)} \\ &= \frac{g_0 + g_1\,\tau}{N} \quad , \end{aligned}$$

find κ via

$$\kappa := \mathtt{Round}\left(\frac{g_0}{N} + \frac{g_1}{N}\,\tau\right) \quad , \tag{61}$$

and obtain ρ via

$$\rho := \gamma - \kappa\,\delta \quad .$$

The following algorithm gives these steps explicitly in terms of the coefficients.

Routine 62 *(Division in $\mathbb{Z}[\tau]$)*

Input:
the dividend $\gamma = c_0 + c_1\,\tau$ and divisor $\delta = d_0 + d_1\,\tau$

Output:

the quotient $\kappa = q_0 + q_1\,\tau$ *and the remainder* $\rho = r_0 + r_1\,\tau$

Computation:

$g_0 \leftarrow c_0\,d_0 + \mu\,c_0\,d_1 + 2\,c_1\,d_1$
$g_1 \leftarrow c_1\,d_0 - c_0\,d_1$
$N \leftarrow d_0^2 + \mu\,d_0\,d_1 + 2\,d_1^2$
$\lambda_0 \leftarrow g_0/N$
$\lambda_1 \leftarrow g_1/N$
$(q_0, q_1) \leftarrow \texttt{Round}(\lambda_0, \lambda_1)$
$r_0 \leftarrow c_0 - d_0\,q_0 + 2\,d_1\,q_1$
$r_1 \leftarrow c_1 - d_1\,q_0 - d_0\,q_1 - \mu\,d_1\,q_1$
`Output` q_0, q_1, r_0, r_1

If we disregard the quotient κ and only output the remainder ρ, this routine may be viewed as a modular reduction algorithm. In this case, we write

$$\rho := \gamma \bmod \delta \ .$$

In analogy with (55) we have

$$\rho = \delta\,\left(\!\!\left(\frac{\gamma}{\delta}\right)\!\!\right) \ . \tag{63}$$

It follows via the first item of (59) that the remainder as we have defined it does indeed have norm as small as possible. It follows via the second item of (59) that

$$N(\rho) \leq \frac{4}{7}\,N(\delta) \ . \tag{64}$$

This represents a strengthening of the ordinary Euclidean condition (26) for this particular ring.

6 Reduced τ-adic NAF's

Having developed the modular reduction operation in $\mathbb{Z}[\tau]$, we now define the reduced τ-adic NAF and apply it to the problem of efficient elliptic scalar multiplication.

6.1 Equivalence of τ-adic NAF's

We recall from the discussion at the end of §4.3 that our goal is a reduced τ-adic NAF for n, equivalent to the ordinary τ-adic NAF for n but only half as long.

We first define what we mean by *equivalent*. Let $\mathcal{G}$ be a set of points on an ABC, and suppose that γ and ρ are two elements of $\mathbb{Z}[\tau]$ for which $\gamma P = \rho P$ for all $P \in \mathcal{G}$. Then we say that TNAF(γ) and TNAF(ρ) are *equivalent* with respect to $\mathcal{G}$. The terminology comes from the fact either TNAF(γ) or TNAF(ρ) can be used to multiply a point in $\mathcal{G}$ by γ.

The following result of [MS93] gives a simple condition for two τ-adic NAF's to be equivalent with respect to the entire set $\mathcal{G} := E_a(\mathbb{F}_{2^m})$ of $\mathbb{F}_{2^m}$-rational points on E.

Proposition 65 *If γ and ρ are elements of $\mathbb{Z}[\tau]$ with*

$$\gamma \equiv \rho \pmod{\tau^m - 1} \ , \tag{66}$$

then

$$\gamma P = \rho P$$

for all $P \in E_a(\mathbb{F}_{2^m})$. Thus TNAF($\gamma$) *and* TNAF($\rho$) *are equivalent with respect to $E_a(\mathbb{F}_{2^m})$.*

Proof. Applying the mapping (15) m times to $P := (x, y)$, we obtain

$$\tau^m P = (x^{2^m}, y^{2^m}) \ .$$

But $x^{2^m} = x$ and $y^{2^m} = y$ since x and y are elements of $\mathbb{F}_{2^m}$. Thus

$$\tau^m P = P$$

for all $P \in E_a(\mathbb{F}_{2^m})$. It follows that

$$(\tau^m - 1) P = \mathcal{O} \tag{67}$$

for all $P \in E_a(\mathbb{F}_{2^m})$.

Suppose now that (66) holds. Then

$$\gamma = \rho + \kappa \cdot (\tau^m - 1)$$

for some $\kappa \in \mathbb{Z}[\tau]$. Therefore

$$\begin{aligned} \gamma\, P &= \rho\, P + \kappa \cdot (\tau^m - 1)\, P \\ &= \rho\, P + \kappa\, \mathcal{O} \\ &= \rho\, P + \mathcal{O} \\ &= \rho\, P \ , \end{aligned}$$

proving the result. □

6.2 Equivalence for Points of Prime Order

One can sharpen (65) in the case of cryptographic interest, namely the main subgroup in an ABC of very nearly prime order (see §4.1). As in that section, the order of the curve is

$$\# E_a(\mathbb{F}_{2^m}) = f \cdot r \ ,$$

where r is prime and $f = 2$ or 4 according to (11). By (25), the element

$$\delta := (\tau^m - 1)/(\tau - 1) \tag{68}$$

has norm r.

In this section, we will show that one can weaken the hypotheses of (65) and still retain equivalence with respect to the main subgroup.

Proposition 69 *Let P be a point in the main subgroup in an ABC of very nearly prime order, and define δ as in (68). Then*

$$\delta\, P = \mathcal{O} \ .$$

Proof. By (12), there is a point Q such that $P = f\, Q$. By (67), we have

$$(\tau^m - 1)\, Q = \mathcal{O} \ .$$

By (68), it follows that

$$\delta \cdot (\tau - 1)\, Q = \mathcal{O} \ . \tag{70}$$

Applying the operation $\overline{\tau - 1}$ to both sides of (70), we obtain

$$\begin{aligned} \mathcal{O} &= \delta \cdot (\tau - 1) \cdot (\overline{\tau - 1})\, Q \\ &= \delta \cdot N(\tau - 1)\, Q \\ &= \delta \cdot f\, Q \\ &= \delta\, P \ , \end{aligned}$$

proving the result. □

Theorem 3. *Let P be a point in the main subgroup in an ABC of very nearly prime order, and define δ as in (68). If γ and ρ are elements of $\mathbb{Z}[\tau]$ with*

$$\gamma \equiv \rho \pmod{\delta} \ ,$$

then

$$\gamma\, P = \rho\, P \ .$$

Thus $\mathrm{TNAF}(\gamma)$ *and* $\mathrm{TNAF}(\rho)$ *are equivalent with respect to the main subgroup.*

Proof. The result follows from (69) in the same way as (65) follows from (67).

6.3 The Reduced τ-adic NAF

Suppose that $E_a(\mathbb{F}_{2^m})$ has very nearly prime order, and that r is the order of the main subgroup. Let n be a positive integer less than $r/2$, and let δ be as in (68). We define the *reduced τ-adic NAF* of n to be

$$\mathrm{RTNAF}(n) := \mathrm{TNAF}(\rho) \ ,$$

where

$$\rho := n \bmod \delta \ .$$

It follows from Thm. 3 that $\mathrm{RTNAF}(n)$ and $\mathrm{TNAF}(n)$ are equivalent with respect to the main subgroup. Thus $\mathrm{RTNAF}(n)$ can be used in place of $\mathrm{TNAF}(n)$ for elliptic scalar multiplication in the main subgroup. It follows from the next theorem that this is a more efficient choice.

Theorem 4. *The average Hamming weight among reduced* TNAF*'s is* $\sim m/3$.

Proof. Since the Hamming weight of an RTNAF is the product of its length and its density, we estimate both. We begin with the length.

It follows from (21) that

$$r = 2^{m-2+a} - (V_m - 1)\, 2^{a-2} \ ; \tag{71}$$

thus

$$r = 2^{m-2+a} + O(2^{m/2})$$

by (18). Now

$$N(\rho) \leq \frac{4}{7}\, r$$

by (64), so that

$$N(\rho) \leq \frac{2^{m+a}}{7} + O(2^{m/2}) \ .$$

By (53), we may conclude that the length ℓ_{RTNAF} of RTNAF(n) satisfies

$$\ell_{\mathrm{RTNAF}} < m + a + .7082392 \ .$$

Since ℓ_{TNAF} is an integer, it follows that

$$\ell_{\mathrm{RTNAF}} \leq m + a \ . \tag{72}$$

We now consider the density of a RTNAF. B. Poonen has outlined a proof[7] that the TNAF's of integers modulo $\tau^m - 1$ have average density $1/3 + o(1)$ as m increases. The proof is easily modified to the case of RTNAF's. The result now follows via (72). □

It follows from (54) that the RTNAF has about half the weight of the ordinary τ-adic NAF. By (8), the weight of RTNAF(n) is about equal to that of NAF(n). Thus, replacing NAF(n) by RTNAF(n) eliminates the elliptic doubles and keeps roughly constant the number of elliptic additions. We have therefore solved the difficulty mentioned at the end of §4.3.

[7] A brief summary of Poonen's approach is given in [Gor98] by D. Gordon, who has since presented a more detailed version of the proof.

7 Elliptic Scalar Multiplication on ABC's

We have now identified the procedure to use for elliptic scalar multiplication on an ABC, namely the analogue of the binary method using the RTNAF. We now present the explicit algorithms for computing the RTNAF, and give the elliptic scalar multiplication algorithms. Finally, we develop the τ-adic analogue of the window method.

7.1 Computing the Reduced τ-adic NAF

To give an algorithm for computing the reduced τ-adic NAF, we need only specialize the modular reduction algorithm (62) to the case of reducing an integer n modulo δ.

Thus we set $\gamma := n$, and let $\delta = d_0 + d_1\,\tau$ be the norm-r element given by (68). Then the integers g_i appearing in (62) are

$$g_i := s_i\,n \ ,$$

where

$$s_0 := d_0 + \mu\,d_1$$
$$s_1 := -d_1 \ .$$

The integers s_i can be expressed in terms of the Lucas sequence U_k via

$$s_i = \frac{(-1)^i}{f}\,(1 - \mu\,U_{m+3-a-i}) \ , \tag{73}$$

where f is as in (11) and μ is as in (14). Since the Lucas sequence U_k can be computed efficiently, so can the integers s_i. They need only be computed once per curve. Once that is done, the reduction method is as follows.

Routine 74 *(Reduction modulo $(\tau^m - 1)/(\tau - 1)$)*

Per-Curve Parameters:
$m,\ a,\ s_0,\ s_1,\ r$

Input:

n

Output:

integers r_0, r_1 *specifying* $r_0 + r_1\,\tau := n \bmod (\tau^m - 1)/(\tau - 1)$

Computation:

$d_0 \leftarrow s_0 + \mu\, s_1$
$\lambda_0 \leftarrow s_0\, n/r$
$\lambda_1 \leftarrow s_1\, n/r$
$(q_0, q_1) \leftarrow \texttt{Round}(\lambda_0, \lambda_1)$ *(via (60))*
$r_0 \leftarrow n - d_0\, q_0 - 2\, s_1\, q_1$
$r_1 \leftarrow s_1\, q_0 - s_0\, q_1$
`Output` r_0, r_1

Note that one could store d_0 rather than computing it during each reduction. This would save one integer addition per reduction, but require an additional $\sim m/2$ bits of memory.

It will be helpful to have a geometric description of the the elements ρ resulting from reducing modulo δ the integers n with $0 \le n < r$. Following [Gor98], we define the *Voronoi region*

$$\mathcal{V} := \{\delta\,\lambda \,:\, \lambda \in \mathcal{U}\} \ .$$

More explicitly, $\mathcal{V}$ is given by the equations

$$\begin{aligned} -r \le\ & (2\,d_0 + \mu\,d_1)\,\lambda_0 + (\mu\,d_0 + 4\,d_1)\,\lambda_1 \ < r \\ -2r \le\ & (d_0 + 4\,\mu\,d_1)\,\lambda_0 - (3\,\mu\,d_0 - 2\,d_1)\,\lambda_1 < 2r \\ -2r \le\ & (d_0 - 3\,\mu\,d_1)\,\lambda_0 + (4\,\mu\,d_0 + 2\,d_1)\,\lambda_1 < 2r \ . \end{aligned}$$

Proposition 75 *The lattice points*

$$\lambda_0 + \lambda_1\,\tau, \quad \lambda_i \in \mathbb{Z}$$

in $\mathcal{V}$ *are precisely the elements*

$$n \bmod \delta, \quad 0 \le n < r \ . \tag{76}$$

Proof. It follows from (63) that every element (76) must be a lattice point in $\mathcal{V}$. We must now prove the converse, *i.e.* that every lattice point in $\mathcal{V}$ is the result of reducing modulo δ an integer n.

We know that every lattice point in $\mathcal{V}$ is the result of reducing modulo δ an element of $\mathbb{Z}[\tau]$. Since the lattice points are incongruent modulo δ, this means that they correspond to the elements of $\mathbb{Z}[\tau]/\delta\,\mathbb{Z}[\tau]$. Now, since δ has norm r, then

$$\frac{\mathbb{Z}[\tau]}{\delta\,\mathbb{Z}[\tau]} \cong \frac{\mathbb{Z}}{r\,\mathbb{Z}} \ .$$

Thus there are precisely r lattice points in $\mathcal{V}$. Finally, since

$$\mathbb{Z} \cap \delta\,\mathbb{Z}[\tau] = r\,\mathbb{Z}$$

(see [ST87]), the integers $0, 1, \ldots, r-1$ are incongruent modulo δ; thus the r elements (76) are distinct. Thus each of the r lattice points in $\mathcal{V}$ is an element (76) for some n. □

Finally, we combine the modular reduction routine (74) with Alg. 1 to obtain the following algorithm for computing the reduced τ-adic NAF.

Algorithm 2 *(Reduced τ-adic NAF)*

Parameters:
$m,\ a,\ s_0,\ s_1,\ r$

Input:
a positive integer n

Output:
RTNAF(n)

Computation:
`Compute` $\rho \leftarrow n \bmod \delta$ *(via (74))*
`Compute` $\mathcal{S} \leftarrow \mathrm{TNAF}(\rho)$ *(via Alg. 1)*
`Output` $\mathcal{S}$

7.2 The τ-adic Method for Elliptic Scalar Multiplication

We now apply the reduced τ-adic NAF to produce an efficient procedure for performing elliptic scalar multiplication on the main subgroup of $E_a(\mathbb{F}_{2^m})$. This is done by modifying Alg. 2 in the same way as was done for (4). This produces the following analogue of (6).

Algorithm 3 *Scalar Multiplication on ABC's*

Per-Curve Parameters:
m, a, s_0, s_1, r

Input:
n, a positive integer less than $r/2$
P, a point in the main subgroup

Output:
nP

Computation:
Compute $(r_0, r_1) \leftarrow n \bmod \delta$ *(via (74))*
Set $Q \leftarrow \mathcal{O}$
$P_0 \leftarrow P$
While $r_0 \neq 0$ or $r_1 \neq 0$
If r_0 odd then
set $u \leftarrow 2 - (r_0 - 2\, r_1 \bmod 4)$
set $r_0 \leftarrow r_0 - u$
if $u = 1$ then set $Q \leftarrow Q + P_0$
if $u = -1$ then set $Q \leftarrow Q - P_0$
Set $P_0 \leftarrow \tau P_0$ (=RightShift[P_0])
Set $(r_0, r_1) \leftarrow (r_1 + \mu\, r_0/2, -r_0/2)$
EndWhile
Output Q

Since $\ell \approx m$, then Alg. 3 requires $\sim m/3$ additions and no doubles. This is at least 50% faster than any of the earlier versions, as shown in Table 1. The "length" and "density" columns in Table 1 give the approximate length of the relevant representation of the

Table 1. Comparison of Elliptic Scalar Multiplication Techniques.

Type of Curve	Method	Length of Expansion	Avg. Density	Avg. # of Elliptic Operations
General	Binary Method	m	1/2	$3m/2$
"	Addition-Subtraction (1989)	m	1/3	$4m/3$
ABC	Koblitz, Balanced (1991)	$2m$	3/8	$3m/4$
"	Meier-Staffelbach (1992)	m	1/2	$m/2$
"	τ-adic NAF (1997)	m	1/3	$m/3$

number and the average density of nonzero terms. The density figure of 3/8 for Koblitz' "balanced" expansions is from experimental observation and may be only an approximation.

It should also be noted that the advantage of the three τ-adic methods over the general methods is slightly overstated in Table 1, since doubling is more efficient than adding on a general elliptic curve.

7.3 The Width-w τ-adic NAF

It remains to extend Alg. 3 to a "τ-adic window method" analogous to (10). In order to do this, we first develop a width-w τ-adic NAF.

In order to motivate our method, we recall the strategy used in the ordinary width-w window method. With that method, we precompute and store the points uP as u runs over representatives of each odd congruence class (mod 2^w). When an odd integer c is encountered, the w rightmost bits are examined to determine which congruence class (mod 2^w) contains c. The corresponding point eP is subtracted off, and the new coefficient $c - e$ is divisible by 2^w.

For a τ-adic window method, we precompute and store a point corresponding to each odd congruence class (mod τ^w). (By "odd" is meant that the element $e_0 + e_1\,\tau$ has odd e_0.) When an odd element $r_0 + r_1\,\tau$ is encountered, we must determine which congruence class (mod τ^w) contains $r_0 + r_1\,\tau$. We can then subtract off the representative of that congruence class and produce a new coefficient which is divisible by τ^w.

In analogy with the ordinary case, we can make the determination by examining the w rightmost bits of the appropriate combination of r_0 and r_1. Define the quantities t_k via

$$t_k := 2\, U_{k-1}\, U_k^{-1} \pmod{2^k} \ .$$

Since the Lucas elements U_k are odd, it follows that t_k is a well-defined integer modulo 2^k that is even but not divisible by 4. Therefore

$$t_k^2 - \mu\, t_k + 2 \equiv 0 \pmod{2^k}$$

by (20). Thus t_k satisfies the same polynomial equation over $\mathbb{Z}/2^k\mathbb{Z}$ that τ satisfies over the complex numbers. It follows that the correspondence $\tau \mapsto t_k$ induces a ring homomorphism from $\mathbb{Z}[\tau]$ onto $\mathbb{Z}/2^k\mathbb{Z}$ via

$$\begin{aligned} \phi_k : \qquad & \mathbb{Z}[\tau] \longrightarrow \mathbb{Z}/2^k\mathbb{Z} \\ & u_0 + u_1\,\tau \longmapsto u_0 + u_1\, t_k \ . \end{aligned}$$

It is easy to see that the odd elements of $\mathbb{Z}[\tau]$ correspond precisely to the odd elements of $\mathbb{Z}/2^k\mathbb{Z}$.

We now compute the kernel of the mapping ϕ_k.

Lemma 77 *Let $\alpha \in \mathbb{Z}[\tau]$. Then $\phi_k(\alpha) = 0$ if and only if α is divisible by τ^k.*

Proof. Since $\phi_k(\tau) = t_k \equiv 2 \pmod 4$, it follows that $\phi_k(\tau^j) = t_k^j$ is divisible by 2^j but not by 2^{j+1}. Thus the power of 2 dividing $\phi_k(\alpha)$ is the power of τ dividing α. □

It follows that each congruence class $(\mathrm{mod}\ \tau^k)$ in $\mathbb{Z}[\tau]$ corresponds under ϕ_k with an element of $\mathbb{Z}/2^k\mathbb{Z}$. Moreover, the odd congruence classes $(\mathrm{mod}\ \tau^k)$ in $\mathbb{Z}[\tau]$ correspond under ϕ_k with the odd elements of $\mathbb{Z}/2^k\mathbb{Z}$.

We now apply these results to construct a τ-adic width-w window method. It follows from the above correspondence that the odd numbers

$$\pm 1,\ \pm 3, \ldots, \pm(2^{w-1} - 1) \tag{78}$$

are incongruent modulo τ^w. We can therefore precompute and store the points uP for each odd u with $2 < u < 2^{w-1}$. The congruence class that contains a given odd element $u_0 + u_1\,\tau$ is the one corresponding to $u_0 + u_1\,t_w \pmod{2^w}$.

We thus have a technique for generating a width-w τ-adic NAF. If we apply it to n modulo δ, we obtain the *reduced width-w τ-adic NAF* of n, or $\mathrm{RTNAF}_w(n)$. The algorithm for producing it is as follows.

Algorithm 4 *(Reduced Width-w τ-adic NAF)*

Parameters:

$m,\ a,\ s_0,\ s_1,\ r,\ t_w$

Input:

a *positive integer* n

Output:

$\mathrm{RTNAF}_w(n)$

Computation:

```
Compute (r0, r1) ← n mod δ      (via (74))
Set S ← ⟨ ⟩
While r0 ≠ 0 or r1 ≠ 0
    If r0 odd
        then
            set u ← r0 + r1 tw mods 2^w
            set r0 ← r0 − u
        else
            set u ← 0
    Prepend u to S
    Set (r0, r1) ← (r1 + μ r0/2, −r0/2)
EndWhile
Output S
```

We now modify Alg. 4 to produce a τ-adic window method for elliptic scalar multiplication.

Algorithm 5 *τ-adic Width-w Window Method*

Per-Curve Parameters:

m, a, s_0, s_1, r, t_w

Input:

a *positive integer* n
an *elliptic curve point* P

Output:

the *point* nP

Precomputation:

Set $P_0 \leftarrow P$
Set $P_{2^{w-2}-1} \leftarrow 2\,P_0$
For i from 1 to $2^{w-2}-1$ do
    Set $P_i \leftarrow P_{i-1} + P_{2^{w-2}-1}$
Next i

Computation:

Set $j \leftarrow 0$
Compute $(r_0, r_1) := n \bmod \delta$ *(via (74))*
Set $Q \leftarrow \mathcal{O}$
While $r_0 \neq 0$ or $r_1 \neq 0$
    If r_0 odd then
        set $u \leftarrow r_0 + r_1\,t_w \bmod 2^w$
        set $r_0 \leftarrow r_0 - u$
        set $i \leftarrow (|u| - 1)/2$
        if $u > 0$
            then
                set $Q \leftarrow Q + P_i$
            else
                set $Q \leftarrow Q - P_i$
    Set $j \leftarrow j + 1$
    Set $Q \leftarrow \tau^{-1}Q$ =LeftShift[Q]
    Set $(r_0, r_1) \leftarrow (r_1 + \mu\, r_0/2, -r_0/2)$

```
EndWhile
    Set Q ← τ^j Q          =RightShift[Q] (j times)
Output Q
```

Note that the expression $(|u| - 1)/2$ is evaluated by dropping the rightmost bit of $|u|$.

Alg. 5 is a right-to-left window algorithm. We saw in §3.2 that this is usually impossible, but it can be done in the case of ABC's because the Frobenius map τ can be inverted efficiently.

One can choose representatives other than (78) for the odd congruence classes (mod τ^w). For example, one can choose the set of all elements

$$\pm \left(1 + \sum_{i=1}^{w-2} u_i \, \tau^i \right)$$

with each $u_i = 0$ or 1. With this choice, the precomputation can be done with one fewer elliptic addition than with (78), although it is necessary to use a table lookup (with 2^{w-2} entries) to assign the precomputed points to their proper addresses.

7.4 Performance

Performing Alg. 5 (or, rather, the variant just mentioned) requires enough memory to store $2^{w-2} + 2$ points. The precomputation requires $2^{w-2} - 1$ elliptic additions; the main computation is performed on a sequence of length $\sim m$ and average density $(w+1)^{-1}$. It follows that a scalar multiplication on $E_a(\mathbb{F}_{2^m})$ requires

$$\sim 2^{w-2} - 1 + \frac{m}{w+1} \tag{79}$$

elliptic additions on average.

Table 2 gives the performance of this algorithm on the curve $E_1(\mathbb{F}_{2^{163}})$ for various widths, based on the estimate (79). (Entries are rounded to the nearest tenth of an integer.) The case $w = 2$ is just Alg. 3. By choosing $w = 5$, one saves well over one-third the work. For larger w, the precomputation costs overshadow any savings on the real-time computation. Thus such larger widths would only be used for a long-term fixed point P (*e.g.* a public key).

Table 2. Performance at Various Widths.

Width	Number of Elliptic Operations		
	Precomp-utation	Real Time (avg)	Total (avg)
2	0	54.3	54.3
3	1	40.8	41.8
4	3	32.6	35.6
5	7	27.2	34.2
6	15	23.3	38.3
7	31	20.4	51.4

It is remarkable that one can perform a general elliptic scalar multiplication on $E_1(\mathbb{F}_{2^{163}})$ using only about 34 multiplicative inversions and 68 field multiplications.

One could obtain still further speedups by using more general window methods. These would be straightforward adaptations of existing methods such as those found in [KT93]. On the other hand, such methods are less automatic than the above fixed-width-window technique, so that more complicated up-front calculations are needed.

8 Efficient Modular Reduction

The modular reduction algorithm (74) is central to computing the reduced τ-adic NAF, but it can be expensive to implement. The main computational difficulty is in the divisions necessary in computing the expression (61). The purpose of this section is to present an equivalent method which is far more efficient.

8.1 Partial Modular Reduction

The equivalent method involves replacing the modular reduction process (74) by a simplified technique called *partial modular reduction.* As a result, one obtains an element $\rho' \in \mathbb{Z}[\tau]$ that is congruent to n modulo δ, but not necessarily of minimal norm.

This is done by replacing the rational numbers λ_i by approximations λ_i' which can be computed more efficiently. We denote by C

the number of bits of accuracy of the approximations λ_i'. The larger C is, the more work is required in computing λ_i', but the greater the likelihood that λ_i' will actually equal λ_i.

We begin by presenting the algorithm for computing λ_i'. We call this process *approximate division* because it replaces the division used to compute $s_i\, n/r$ by two K-bit multiplications, where

$$K := \frac{m+5}{2} + C \ . \tag{80}$$

Algorithm 6 *(Approximate Division by r)*

Per-Curve Parameters:

s_i, r
V_m *(see §4.1)*

Input:

a *positive integer* n *less than* $r/2$

Output:

$\lambda_i := s_i\, n/r$ *to* C *bits of accuracy*

Computation:

1 $\quad n' \leftarrow \left\lfloor \frac{n}{2^{m-K-2+a}} \right\rfloor$

2 $\quad g' \leftarrow s_i\, n'$

3 $\quad h' \leftarrow \left\lfloor \frac{g'}{2^m} \right\rfloor$

4 $\quad j' \leftarrow V_m\, h'$

5 $\quad \ell' \leftarrow \texttt{Round}\left(\frac{g'+j'}{2^{K-C}}\right)$

6 $\quad \texttt{Output}\ \ \lambda' := \ell' \,/\, 2^C \ .$

When the output of Alg. 6 is used in the modular reduction process (74), the resulting algorithm is what we call *partial reduction* modulo $\delta := (\tau^m - 1)/(\tau - 1)$. We shall write

$$\rho' = n \operatorname{partmod} \delta$$

for the result. The explicit algorithm is as follows.

Algorithm 7 *(Partial Reduction modulo $(\tau^m - 1)/(\tau - 1)$)*

Per-Curve Parameters:
m, a, s_0, s_1, r

Input:
n

Output:
integers r'_0, r'_1 *specifying* $r'_0 + r'_1 \tau := n \operatorname{partmod} \delta$

Computation:
$d_0 \leftarrow s_0 + \mu s_1$
$\lambda'_0 \leftarrow s_0 n/r$ *to* $\sim K$ *places via Alg. 6*
$\lambda'_1 \leftarrow s_1 n/r$ *to* $\sim K$ *places via Alg. 6*
$(q'_0, q'_1) \leftarrow \texttt{Round}(\lambda'_0, \lambda'_1)$
$r'_0 \leftarrow n - d_0 q'_0 - 2 s_1 q'_1$
$r'_1 \leftarrow s_1 q'_0 - s_0 q'_1$
`Output` r'_0, r'_1

We define the *partially reduced τ-adic NAF* of n by

$$\mathrm{PRTNAF}(n) := \mathrm{TNAF}(\rho') \ .$$

The element ρ' is congruent to n modulo δ, since

$$\rho' = n - \kappa' \delta$$

where $\kappa' := q'_0 + q'_1 \tau$. Therefore, $\mathrm{PRTNAF}(n)$ and $\mathrm{TNAF}(n)$ are equivalent with respect to the main subgroup. Thus $\mathrm{PRTNAF}(n)$ can be used in place of $\mathrm{TNAF}(n)$ for elliptic scalar multiplication in the main subgroup.

On the other hand, ρ' may not be of minimal norm, as ρ would have been. Thus PRTNAF(n) may be longer than RTNAF(n). If so, then more elliptic operations will be required in the scalar multiplication step. This would mitigate, and in all likelihood wipe out, the savings in avoiding the integer divisions in (74).

We examine this question in the next section. In particular, we shall prove that PRTNAF(n) is never much longer than RTNAF(n). More importantly, we will show that the probability that RTNAF(n) $\neq$ PRTNAF(n) can be made arbitrarily small, so that the possibility need not trouble us in practice.

8.2 Analysis of Partial Modular Reduction

We begin by analyzing the computational requirements of the partial modular reduction process.

Proposition 81 *Partial modular reduction, as implemented by Algs. 6 and 7, requires eight K-bit multiplications and no integer divisions, where K is given by (80).*

Proof. Since Alg. 7 requires two executions of Alg. 6 (one for each i), a total of eight multiplications is required. No integer divisions are required since all denominators are powers of 2, so that the required divisions are implemented by simple bit shifts.

The only remaining issue is the number of bits in the numbers to be multiplied. Those numbers are n', h', V_m, d_0, and s_i and q_i' for $i = 0$, 1. Each must be less than 2^K in absolute value.

The desired bound on n' follows from the fact that $n < r/2$. The bounds for s_i, d_0, and V_m come from (18) via (73). From these bounds is derived the bound for h', along with

$$|\lambda_i'| < 2^{(m-1)/2}$$

for each i. Thus $|q_i'| < 2^K$. □

We next address the accuracy of partial modular reduction.

Theorem 5. *If $\lambda_i := g_i/r$, and λ_i' is the approximation obtained by Thm. 6, then*

$$|\lambda_i - \lambda_i'| < 2^{-C} \quad .$$

Proof. It follows from (71) that

$$\frac{g_i}{r} = \frac{g_i}{2^{m-a+2}} + \frac{(V_m - 1)\, g_i}{2^{2m-a+2}} + \frac{(V_m - 1)^2\, g_i}{2^{2m}\, r} \quad ,$$

so that

$$\frac{g_i}{r} = \frac{g_i}{2^{m-a+2}} + \frac{V_m\, g_i}{2^{2m-a+2}} + O(2^{-m/2}) \quad .$$

Therefore

$$\frac{2^K\, g_i}{r} = \frac{g_i}{2^{m-K-a+2}} \left(1 + \frac{V_m}{2^m}\right) + O(2^{K-m/2}) \quad . \tag{82}$$

Now it follows from the definition of g' that

$$\left| \frac{g_i}{2^{m-K-a+2}} - g' \right| < |s_i| \quad .$$

Thus by (82),

$$\frac{2^K\, g_i}{r} = g' + \frac{g'\, V_m}{2^m} + \theta_0\, s_i + O(2^{K-m/2})$$

for some θ_0 with $|\theta_0| < 1$. It follows from the definition of j' that

$$\left| j' - \frac{g'\, V_m}{2^m} \right| < |V_m| \quad .$$

Thus

$$\frac{2^K\, g_i}{r} = g' + j' + \theta_0\, s_i + \theta_1\, V_m + O(2^{K-m/2})$$

for some θ_1 with $|\theta_1| < 1$. It follows that

$$\frac{2^K\, g_i}{r} = g' + j' + \theta\, (|s_i| + |V_m|)$$

for some θ with $|\theta| < 1$. Now

$$\begin{aligned} |s_i| + |V_m| &< 2^{1+m/2} + 2^{(m-1)/2} \\ &< 2^{(m+3)/2} \\ &\le 2^{K-C-1} \quad , \end{aligned}$$

so that

$$\left| \frac{2^K g_i}{r} - (g' + j') \right| < 2^{K-C-1} \ .$$

It follows that

$$\left| \frac{2^C g_i}{r} - \frac{g' + j'}{2^{K-C}} \right| < \frac{1}{2} \ ,$$

so that

$$\left| \frac{2^C g_i}{r} - \ell' \right| < 1 \ .$$

Dividing this equation by 2^C gives the result. □

8.3 Worst-Case Effect of Partial Modular Reduction

In this section and the next, we retain the notation κ, ρ for the results of modular reduction and κ', ρ' for the results of partial modular reduction of n modulo δ.

We have seen that

$$N(\rho) \leq \frac{4}{7} r \ ,$$

and that therefore the TNAF(ρ) has length at most $m + a$. (See Thm. 4.) We now obtain corresponding results for ρ'.

Lemma 83 *If* $\rho' := n \operatorname{partmod} \delta$, *then*

$$N(\rho') \leq \left(\frac{2}{\sqrt{7}} + \sqrt{N(\varepsilon)} \right)^2 r \ ,$$

where $\varepsilon = \kappa - \kappa'$.

Proof. Since

$$\begin{aligned} \rho' - \rho &= (n - \kappa' \delta) - (n - \kappa \delta) \\ &= \varepsilon \delta \ , \end{aligned}$$

then

$$\sqrt{N(\rho')} \leq \sqrt{N(\rho)} + \sqrt{N(\epsilon)} \sqrt{r}$$

by the Triangle Inequality (23). The result then follows from (64). □

Theorem 6. *If* $C \geq 2$, *then the length of* PRTNAF(n) *is at most* $m + a + 3$.

Proof. By Thm. 5,

$$|\lambda_i - \lambda'_i| < \frac{1}{4}$$

for each i. It is easy to deduce that λ and λ' are either in the same copy of $\mathcal{U}$ or adjacent copies. It follows that the centers of those copies satisfy

$$\kappa - \kappa' = 0,\ \pm 1,\ \pm\tau,\ \text{or}\ \pm\overline{\tau}\ .$$

Thus $N(\kappa - \kappa') \leq 2$, so that

$$N(\rho') < 4.7095185\, r$$

by (83). It follows via (53) that the length of $\mathrm{TNAF}(\rho')$ is less than $m + a + B$, where $B = 3.7511737$. The result now follows from the fact that the length must be an integer. □

Thm. 6 states that the upper bound for the length of the PRT-NAF is three bits higher than the corresponding bound for the RT-NAF. In any given instance, the difference in length need not be three bits, since neither TNAF need attain its maximum possible length. Indeed, we will next prove that the two TNAF's are identical almost all of the time.

8.4 Practical Effect of Partial Modular Reduction

Proposition 84 *Let $T > 0$ and $\lambda := g/r$, where g is an integer not divisible by r for which*

$$|g \bmod s\ r| \leq \frac{r}{2} - T\ . \tag{85}$$

(The "mods" notation is as in §3.2.) If z is a rational number with

$$|z - \lambda| < \frac{T}{r}\ ,$$

then

$$\mathtt{Round}(\lambda) = \mathtt{Round}(z)\ .$$

Proof. The inequality (85) can be rewritten as

$$| \, (\!(\lambda)\!) \, | \leq \frac{1}{2} - \frac{T}{r} \ .$$

If $\kappa = \texttt{Round}(\lambda)$, then,

$$|\lambda - \kappa| \leq \frac{1}{2} - \frac{T}{r} \ .$$

Therefore

$$\begin{aligned} |z - \kappa| &\leq |z - \lambda| + |\lambda - \kappa| \\ &< \frac{T}{r} + \left(\frac{1}{2} - \frac{T}{r} \right) \\ &\leq \frac{1}{2} \ , \end{aligned}$$

so that $\texttt{Round}(z) = \kappa$. □

Corollary 86 *Let $T > 0$, and let s be an integer not divisible by r. If ε is a real number with*

$$|\varepsilon| < \frac{T}{r} \ ,$$

then

$$\texttt{Round}\left(\frac{s\,n}{r}\right) = \texttt{Round}\left(\frac{s\,n}{r} + \varepsilon\right)$$

for all n between 0 and $r/2$, with at most T exceptions.

Proof. By (84), the only exceptions are the values of n for which

$$\frac{r}{2} - T \leq |s\,n \bmod r| < \frac{r}{2} \ .\square$$

The following result is proved similarly to (86).

Proposition 87 *Let $T > 0$, and let s be an integer not divisible by r. If*

$$|\varepsilon| < \frac{T}{r} \ ,$$

then

$$\left\lfloor \frac{s\,n}{r} \right\rfloor = \left\lfloor \frac{s\,n}{r} + \varepsilon \right\rfloor$$

for all n between 0 and $r/2$, with at most T exceptions.

Lemma 88 *Let r be an odd prime, and let s_0 and s_1 be integers not divisible by r. For n an integer between 0 and $r/2$, let $\lambda_i := s_i\, n/r$ for $i = 0,\ 1$. For each i, write*

$$\lambda_i = \lambda_i' + \delta_i$$

where

$$|\delta_i| < \frac{L}{r} \ . \tag{89}$$

Then

$$\texttt{Round}(\lambda_0, \lambda_1) = \texttt{Round}(\lambda_0', \lambda_1')$$

for all values of n with at most $14\,L$ exceptions.

Proof. We examine the quantities f_i, η_i, and h_i which are computed in the course of executing the algorithm (60) on $\lambda_0 + \lambda_1\,\tau$, and compare them to the corresponding quantities f_i', η_i', and h_i' from the same computation on $\lambda_0' + \lambda_1'\,\tau$.

It follows from (86) with $T := L$ and $\varepsilon := \delta_0$ that $f_0 = f_0'$ for all n with at most L exceptions. Similarly, $f_1 = f_1'$ for all n with at most L exceptions. Therefore, we have $f_i = f_i'$ for both i, for all n with at most $2\,L$ exceptions.

For each nonexceptional n, we have

$$\eta_i = \eta_i' + \delta_i$$

for each i. It follows that

$$\begin{aligned} |\eta - \eta'| &= |2\delta_0 + \mu\,\delta_1| \\ &< \frac{3\,L}{r} \ . \end{aligned}$$

It follows from (87) with $T := 3\,L$ and $\varepsilon := \eta - \eta'$ that

$$\lfloor \eta \rfloor = \lfloor \eta' \rfloor$$

for all n with at most $3\,L$ exceptions. For these n,

$$\eta \geq 1 \text{ if and only if } \ \eta' \geq 1$$

and

$$\eta < -1 \text{ if and only if } \eta' < -1 \ .$$

Similarly, one finds that

$$\eta_0 - 3\,\mu\,\eta_1 \geq 1 \text{ if and only if } \eta'_0 - 3\,\mu\,\eta'_1 \geq 1 \ ,$$

$$\eta_0 - 3\,\mu\,\eta_1 < -1 \text{ if and only if } \eta'_0 - 3\,\mu\,\eta'_1 < -1$$

for all n with at most $4\,L$ exceptions; and that

$$\eta_0 + 4\,\mu\,\eta_1 \geq 2 \text{ if and only if } \eta'_0 + 4\,\mu\,\eta'_1 \geq 2 \ ,$$

$$\eta_0 + 4\,\mu\,\eta_1 < -2 \text{ if and only if } \eta'_0 + 4\,\mu\,\eta'_1 < -2$$

for all n with at most $5\,L$ exceptions.

It follows that all of the above conditions hold for all n, with at most $14\,L$ exceptions. For each nonexceptional n, we have $f_i = f'_i$ for both i, and all of the `if` conditions in (60) evaluate the same for the two computations. Thus $h_i = h'_i$ for $i = 0, 1$. It follows that $q_0 = q'_0$ and $q_1 = q'_1$, for each nonexceptional n. □

Theorem 7. *If partial modular reduction with an accuracy of C bits is used, then the probability that $\rho \neq \rho'$ is less than*

$$\texttt{Prob} < 2^{-(C-5)} \ .$$

Proof. Let $L := 2^{-C}\,r$. Then (89) holds by Thm. 5. By (88), the number of n for which $\rho \neq \rho'$ is at most $14\,L$; thus the probability that this happens is at most

$$\frac{14\,L}{r/2} < \frac{1}{2^{C-5}} \ .\square$$

8.5 Conclusion

We now summarize the advantages and disadvantages of partial modular reduction.

If ordinary modular reduction (74) is used, then four K-bit multiplications are required, where $K := (m+3)/2$. Also, two divisions are required, where the dividend is $\sim 3m/2$ bits in length and the divisor is $\sim m$ bits.

Using partial modular reduction (Algs. 6 and 7) with accuracy C has the following effects.

- The two divisions are no longer required.
- Four additional multiplications are required.
- The bit length of the integers to be multiplied is increased by $C + 1$.
- An answer ρ of non-minimal norm can be obtained with probability $< 2^{-(C-5)}$.

One can often choose C large enough that no non-minimal answer will be encountered in practice, at little practical cost. For example, if $m = 163$, then a multiplier of at least 83 bits is required. The likeliest available multiplier sizes are 96 or 128 bits. In the former case, one can choose $C := 12$, for an non-minimal answer at most once every 128 times. In that case, about three bits will be added to the TNAF, for an increase of about 1.8%. Thus the average cost comes to .08%, well worth it to avoid the divisions. For the latter case, one can choose C up to 54, guaranteeing that non-minimal answers will not occur in practice.

9 Recent Developments and Open Problems

Since the initial publication of these results in the Proceedings of CRYPTO '97, the following advances have been made in the study of ABC's.

- The results of [MS93] have recently been generalized [Mül98] to curves defined over fields of 2^d elements for small d. For example, the curves with complex multiplication by $(\pm 1 + \sqrt{-15})/2$ are defined over $\mathbb{F}_{2^2}$. (The results of this paper should also carry over to this more general situation.)
- Analogues of ABC's over fields of small odd characteristic have been studied [Kob98], and representations analogous to the τ-adic NAF (but with even fewer nonzero terms) have been obtained.
- It has been observed ([GLV98], [WZ98]) that the best square-root attack on elliptic curves [OW99] can be modified in the case of ABC's. Rather than working with the points on the curve, one instead works with the cycles under the Frobenius operation. The

resulting algorithm requires fewer steps than in the general case; however, each step is slightly more expensive.

- Finally, C. Günther and A. Stein have extended the concepts in this paper to the case of hyperelliptic curves [GS99].

The following are open problems in this field.

- We have presented worst-case upper bounds for the length of $\mathrm{TNAF}(\alpha)$ in terms of $N(\alpha)$. It may be more useful to possess analogous average-case results. For example, let

$$F(\alpha) := \ell(\alpha) - \log_2(N(\alpha))$$

  for $\alpha \in \mathbb{Z}[\tau]$, where $\ell(\alpha)$ denotes the length of $\mathrm{TNAF}(\alpha)$. It follows from (53) that

$$-.5462682713 < F(\alpha) < 3.51559412$$

  whenever $\ell(\alpha) > 30$. It would be a nice result to prove the existence of an average value for $F(\alpha)$ and to evaluate it. (Note that the examples at the end of §4.3 prove that there is no asymptotic value for $F(\alpha)$ as $N(\alpha)$ gets large.)

- It is often required (*e.g.* in many cryptographic algorithms such as the ECDSA digital signature [JM99]) to take a random multiple of a point on an elliptic curve. More precisely, let P be in the main subgroup of an ABC of very nearly prime order. To take a random multiple of P, one generates a random integer n modulo r, reduces it modulo the element δ given in (68), and uses the result ρ to compute ρP. The techniques in this paper provide a particularly efficient way of doing the latter, but it would be cheaper still to produce a random point in the main subgroup directly. This could be done[8] by generating a "random τ-adic NAF" of length m.

  By a "random τ-adic NAF" we mean a sequence of 0's and ± 1's which is generated as follows. Generate the first signed bit according to the following probability distribution:

$$u := \begin{cases} 0 & \text{prob} = 1/2 \\ 1 & \text{prob} = 1/4 \\ -1 & \text{prob} = 1/4 \ . \end{cases} \tag{90}$$

[8] This is an adaptation of an idea of H. Lenstra (see [Kob91]).

To generate subsequent signed bits, follow each ± 1 by a 0, and generate each signed bit following a 0 according to (90).

The sequences generated in this way represent random selections from the set of all τ-adic NAF's of given length. In particular, each signed bit occurs with the proper average frequency.[9]

Once the sequence is generated, one computes αP where α is the element represented by the sequence.

This method gives random points, but their distribution is not known. It would be useful to measure how uniformly distributed such points are in the main subgroup.

More precisely, we know from Thm. 4 that every $n < r/2$ has a reduced τ-adic NAF of length at most $m + a$. Thus every point in the main subgroup can be obtained by scalar multiplication using some nonadjacent sequence of $m + a$ signed bits. It is easy to see that the number of such sequences is the integer closest to $2^{m+a+2}/3$. Since there are $r \approx 2^{m-2+a}$ points in the main subgroup, the average number of sequences leading to a given point is 16/3. It would be useful to know how much deviation there is from this average.

Acknowledgement: the author wishes to thank Dan Gordon, Neal Koblitz, Julio Lopez, and Bob Reiter for many helpful comments and suggestions.

References

[ABV89] D. W. Ash, I. F. Blake, and S. Vanstone, "Low complexity normal bases," *Discrete Applied Math.* **25** (1989), pp. 191–210.

[Ber84] E. Berlekamp, *Algebraic Coding Theory*, Aegean Park Press, 1984, pp. 36-44.

[GLV98] R. Gallant, R. Lambert, and S. Vanstone, "Improving the parallelized Pollard lambda search on binary anomalous curves," `http://grouper.ieee.org/groups/1363/contributions/attackABC.ps`

[9] It was proved in (32) that 0 occurs with frequency 2/3 after the initial ± 1. It is easy to see that 1 and -1 are equally likely on average.

[Gor98] D. Gordon, "A survey of fast exponentiation methods," *J. Algs.* **27** (1998), pp. 129-146.

[GS99] C. Günther and A. Stein, to appear.

[IEEE99] Institute of Electrical and Electronics Engineers, *IEEE P1363: Standard Specifications for Public-Key Cryptography*, Draft, 1999.

[ITT86] T. Itoh, O Teechai, and S. Trojii, "A fast algorithm for computing multiplicative inverses in $GF(2^t)$," *J. Soc. Electron. Comm.* (Japan) **44** (1986), pp. 31–36.

[JM99] D. Johnson and A. Menezes, "The Elliptic Curve Digital Signature Algorithm (ECDSA)," Univ. of Waterloo, 1999, `http://cacr.math.waterloo.ca`

[Knu81] D. E. Knuth, *Seminumerical Algorithms*, Addison-Wesley, 1981, p. 272.

[Kob91] N. Koblitz, "CM curves with good cryptographic properties," *Proc. Crypto '91*, Springer-Verlag, 1992, pp. 279–287.

[Kob94] N. Koblitz, *A Course of Number Theory and Cryptography*, 2nd ed., Springer-Verlag, 1994. pp. 327–337.

[Kob98] N. Koblitz, "An elliptic curve implementation of the Finite Field Digital Signature Algorithm," *Proc. Crypto '98*, Springer-Verlag, 1998, pp. 327–337.

[KT93] K. Koyama and Y. Tsuruoka, "Speeding up elliptic cryptosystems by using a signed binary window method," *Proc. Crypto '92*, Springer-Verlag, 1993, pp. 345–357.

[Lop99] J. Lopez, "Fast multiplication on elliptic curves over $GF(2^m)$ without precomputation," preprint.

[MO90] F. Morain and J. Olivos, "Speeding up the computations on an elliptic curve using addition-subtraction chains," *Inform. Theor. Appl.* **24** (1990), pp. 531–543.

[MOV91] A. Menezes, T. Okamoto and S. Vanstone, "Reducing elliptic curve logarithms to logarithms in a finite field," *Proc. 23rd Annual ACM Symp. on Theory of Computing* (1991), pp. 80–89.

[MOV97] A. Menezes, P. van Oorschot, and S. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1997, pp. 107–109.

[MS93] W. Meier and O. Staffelbach, "Efficient multiplication on certain nonsupersingular elliptic curves," *Proc. Crypto '92*, Springer-Verlag, 1993, pp. 333–344.

[Mül98] V. Müller, "Fast multiplication on elliptic curves over small fields of characteristic two," *J. Crypt.* **11** (1998), pp. 219–234.

[OW99] P. van Oorschot and M. Weiner, "Parallel collision search with cryptanalytic applications," *J. Crypt.*, **12** (1999).

[Ser98] G. Seroussi, "Compact representations of elliptic curve points over $GF(2^n)$," `http://grouper.ieee.org/groups/1363/contributions/hp.ps`

[Sil86] J. Silverman, *The Arithmetic of Elliptic Curves*, Springer-Verlag, 1986.

[ST87] I. Stewart and D. Tall, *Algebraic Number Theory*, 2nd. ed. Chapman and Hall, 1987.

[WZ98] M. Weiner and R. Zuccherato, "Faster attacks on elliptic curve cryptosystems," `http://grouper.ieee.org/groups/1363/contributions/attackEC.ps`

A Main Subgroup Membership

In this appendix we state and prove the conditions that are used in §4.1 to determine when a point on an ABC of very nearly prime order is an element of the main subgroup. We work on a general Weierstrass curve

$$E: \quad y^2 + x\,y = x^3 + a\,x^2 + b$$

over $GF(2^m)$. It should be recalled that the order of $E(GF(2^m))$ is always even, and divisible by 4 if and only if a has trace 0.

Proposition 91 *[Ser98] Let (x_2, y_2) be a point on E. Then*

$$(x_2, y_2) = 2\,(x_1, y_1) \tag{92}$$

for some (x_1, y_1) on E if and only if $\mathrm{Tr}(x_2) = \mathrm{Tr}(a)$.

Proof. Suppose first that (92) holds for some (x_1, y_1). Then

$$x_2 = \lambda^2 + \lambda + a$$

where

$$\lambda = x_1 + \frac{y_1}{x_1}\ . \tag{93}$$

Thus

$$\begin{aligned}\mathrm{Tr}(x_2) &= \mathrm{Tr}(\lambda^2) + \mathrm{Tr}(\lambda) + \mathrm{Tr}(a)\\ &= \mathrm{Tr}(a)\end{aligned}$$

since $\mathrm{Tr}(\lambda^2) = \mathrm{Tr}(\lambda)$.

Conversely, suppose now that $\mathrm{Tr}(x_2) = \mathrm{Tr}(a)$. Then $x_2 + a$ has trace 0, so that

$$x_2 + a = \lambda^2 + \lambda \tag{94}$$

for some $\lambda \in GF(2^m)$. Define $x_1 \in GF(2^m)$ by

$$y_2 = x_1^2 + (\lambda + 1)\,x_2\ . \tag{95}$$

(This condition specifies x_1 since every element of $GF(2^m)$ has a unique square root.) Then

$$y_2^2 = x_1^4 + (\lambda^2 + 1)\,x_2^2\ ,$$

and

$$x_2\, y_2 = (\lambda + 1)\, x_1^2\, x_2 \ .$$

Thus

$$\begin{aligned} y_2^2 + x_2\, y_2 &= (\lambda^2 + \lambda)\, x_2^2 + x_1^4 + x_1^2\, x_2 \\ &= (x_2 + a)\, x_2^2 + x_1^4 + x_1^2\, x_2 \end{aligned}$$

by (94). Since (x_2, y_2) satisfies the equation for E, it follows that

$$x_1^4 + x_1^2\, x_2 = b \ . \tag{96}$$

Now define

$$y_1 := \lambda\, x_1 + x_1^2 \ . \tag{97}$$

Then

$$\begin{aligned} y_1^2 &= \lambda^2\, x_1^2 + x_1^4 \\ &= (\lambda + x_2 + a)\, x_1^2 + x_1^4 \ , \end{aligned}$$

and

$$x_1\, y_1 = \lambda\, x_1^2 + x_1^3 \ .$$

Then

$$\begin{aligned} y_1^2 + x_1\, y_1 &= x_1^3 + a\, x_1^2 + (x_1^4 + x_1^2\, x_2) \\ &= x_1^3 + a\, x_1^2 + b \end{aligned}$$

by (96), so that (x_1, y_1) is a point on E. It follows from (97) that (93) holds; combining this with (95), we obtain $(x_2, y_2) = 2\,(x_1, y_1)$. □

Proposition 98 *Suppose that a has trace 0. Let (x_4, y_4) be a point on E. Then*

$$(x_4, y_4) = 4\,(x_1, y_1) \tag{99}$$

for some (x_1, y_1) on E if and only if $\mathrm{Tr}(x_4) = 0$ and

$$\mathrm{Tr}(y_4) = \mathrm{Tr}(\lambda\, x_4) \tag{100}$$

for some λ satisfying

$$\lambda^2 + \lambda = x_4 + a \ . \tag{101}$$

Proof. Suppose first that (99) holds for some (x_1, y_1). Let

$$(x_2, y_2) := 2\,(x_1, y_1) \ ,$$

so that $(x_4, y_4) = 2\,(x_2, y_2)$. By (91), it follows that

$$\mathrm{Tr}(x_4) = \mathrm{Tr}(x_2) = 0 \ . \tag{102}$$

Let

$$\lambda := x_2 + \frac{y_2}{x_2} \ ; \tag{103}$$

then it follows from the doubling formula that (101) holds and that

$$y_4 := x_2^2 + (\lambda + 1)\, x_4 \ . \tag{104}$$

Since $\mathrm{Tr}(x_2^2) = \mathrm{Tr}(x_2)$, it follows (via (102)) that (100) holds.

Conversely, suppose now that $\mathrm{Tr}(x_4) = 0$, and that (100) holds for some λ for which (101) holds. Then in fact (100) holds for *either* value of λ satisfying (101). It follows from (91) that $(x_4, y_4) = 2\,(x_2, y_2)$ for some (x_2, y_2) on E. We may conclude that (103) holds (where we have replaced λ by $\lambda + 1$ if necessary). Moreover, (104) holds, so that

$$y_4 + \lambda\, x_4 = x_2^2 + x_4 \ .$$

It follows that $\mathrm{Tr}(x_2^2) = 0$, so that $\mathrm{Tr}(x_2) = 0$. By (91), it follows that $(x_2, y_2) = 2\,(x_1, y_1)$ for some (x_1, y_1) on E. Thus $(x_4, y_4) = 4\,(x_1, y_1)$. □

B Gaussian Normal Bases

In anticipation of the specific ABC's presented in Appendix C, we describe the Gaussian normal basis representation for the binary field $GF(2^m)$.

Let m be prime, and let T be an even positive integer for which $p := T\,m + 1$ is prime. Then a Type T Gaussian normal basis exists

for $GF(2^m)$ if and only if m is relatively prime to $(p-1)/k$, where k is the order of 2 modulo p. For each prime m, a Type T Gaussian normal basis exists for some T. To maximize efficiency, the lowest available value of T is used.

Once the type T has been identified, the multiplication rule can be constructed. One first constructs a function $F(\underline{u}, \underline{v})$ on inputs

$$\underline{u} = (u_0\, u_1\, \ldots\, u_{m-1})$$

and

$$\underline{v} = (v_0\, v_1\, \ldots\, v_{m-1})$$

as follows. Let u be an integer having order T modulo p. Compute the sequence $J(1)$, $J(2), \ldots, J(p-1)$ as follows:

```
1              Set w ← 1
2              For j from 0 to T − 1 do
  2.1              Set n ← w
  2.2              For i from 0 to m − 1 do
      2.2.1            Set J(n) ← i
      2.2.2            Set n ← 2n mod p
  2.3              Set w ← uw mod p
```

Then F is given by the formula

$$F(\underline{u}, \underline{v}) := \sum_{k=1}^{p-2} u_{J(k+1)}\, v_{J(p-k)}.$$

This computation need only be performed once per basis.

Given the function F for $\mathcal{B}$, one computes the product

$$(c_0\, c_1 \ldots c_{m-1}) = (a_0\, a_1 \ldots a_{m-1}) \times (b_0\, b_1 \ldots b_{m-1})$$

as follows.

```
1          Set (u_0 u_1 ... u_{m-1}) ← (a_0 a_1 ... a_{m-1})
2          Set (v_0 v_1 ... v_{m-1}) ← (b_0 b_1 ... b_{m-1})
3          For k from 0 to m − 1 do
  3.1          Compute c_k := F(u, v)
  3.2          Set u ←LeftShift(u) and v ←LeftShift(v),
                   where LeftShift denotes the
                   circular left shift operation.
4          Output c := (c_0 c_1 ... c_{m-1})
```

Example. For the type 4 normal basis for $GF(2^7)$, one has $p = 29$ and $u = 12$ or 17. Thus the values of F are given by

$$\begin{array}{llll} F(1)=0 & F(8)=3 & F(15)=6 & F(22)=5 \\ F(2)=1 & F(9)=3 & F(16)=4 & F(23)=6 \\ F(3)=5 & F(10)=2 & F(17)=0 & F(24)=1 \\ F(4)=2 & F(11)=4 & F(18)=4 & F(25)=2 \\ F(5)=1 & F(12)=0 & F(19)=2 & F(26)=5 \\ F(6)=6 & F(13)=4 & F(20)=3 & F(27)=1 \\ F(7)=5 & F(14)=6 & F(21)=3 & F(28)=0 \end{array}$$

Therefore

$$\begin{aligned} F(\underline{u},\underline{v}) = {} & u_0\,v_1 + u_1\,(v_0+v_2+v_5+v_6) + u_2\,(v_1+v_3+v_4+v_5) \\ & + \; u_3\,(v_2+v_5) + u_4\,(v_2+v_6) + u_5\,(v_1+v_2+v_3+v_6) \\ & + \; u_6\,(v_1+v_4+v_5+v_6). \end{aligned}$$

Thus, if

$$a = (1\,0\,1\,0\,1\,1\,1) \qquad \text{and} \qquad b = (1\,1\,0\,0\,0\,0\,1),$$

then

$$\begin{aligned} c_0 &= F(\,(1\,0\,1\,0\,1\,1\,1),\,(1\,1\,0\,0\,0\,0\,1)\,) = 1, \\ c_1 &= F(\,(0\,1\,0\,1\,1\,1\,1),\,(1\,0\,0\,0\,0\,1\,1)\,) = 0, \\ &\;\;\vdots \\ c_6 &= F(\,(1\,1\,0\,1\,0\,1\,1),\,(1\,1\,1\,0\,0\,0\,0)\,) = 1, \end{aligned}$$

so that $c = ab = (1\,0\,1\,1\,0\,0\,1)$.

C Standard Curves

The following five ABC's appear in the document "Recommended Elliptic Curves for Federal Government Use," issued July 1999 by NIST and available on their website

`http://csrc.nist.gov/encryption` .

Each curve has very nearly prime order. For each curve is given a base point $G = (G_x, G_y)$ generating the main subgroup.

Curve K-163

$a = 1$

$r = 5846006549323611672814741753598448348329118574063$

Polynomial Basis $t^{163} + t^7 + t^6 + t^3 + 1$

$G_x =$ `2 fe13c053 7bbc11ac aa07d793 de4e6d5e 5c94eee8`

$G_y =$ `2 89070fb0 5d38ff58 321f2e80 0536d538 ccdaa3d9`

Type 4 *Normal Basis*

$G_x =$ `0 5679b353 caa46825 fea2d371 3ba450da 0c2a4541`

$G_y =$ `2 35b7c671 00506899 06bac3d9 dec76a83 5591edb2`

Curve K-233

$a = 0$

$r = 3450873173395281893717377931138512760570940988622521263280\backslash$
87024741343

Polynomial Basis $t^{233} + t^{74} + 1$

$G_x =$ 172 32ba853a 7e731af1
29f22ff4 149563a4 19c26bf5 0a4c9d6e efad6126

$G_y =$ 1db 537dece8 19b7f70f
555a67c4 27a8cd9b f18aeb9b 56e0c110 56fae6a3

Type 2 Normal Basis

$G_x =$ 0fd e76d9dcd 26e643ac
26f1aa90 1aa12978 4b71fc07 22b2d056 14d650b3

$G_y =$ 064 3e317633 155c9e04
47ba8020 a3c43177 450ee036 d6335014 34cac978

Curve K-283

$a = 0$

$r = 3885337784451458141838923813647037813284811733793061324295\backslash$
$874997529815829704422603873$

Polynomial Basis $t^{283} + t^{12} + t^7 + t^5 + 1$

$G_x =$ 503213f 78ca4488 3f1a3b81 62f188e5
53cd265f 23c1567a 16876913 b0c2ac24 58492836

$G_y =$ 1ccda38 0f1c9e31 8d90f95d 07e5426f
e87e45c0 e8184698 e4596236 4e341161 77dd2259

Type 6 Normal Basis

$G_x =$ 3ab9593 f8db09fc 188f1d7c 4ac9fcc3
e57fcd3b db15024b 212c7022 9de5fcd9 2eb0ea60

$G_y =$ 2118c47 55e7345c d8f603ef 93b98b10
6fe8854f feb9a3b3 04634cc8 3a0e759f 0c2686b1

Curve K-409

$a = 0$

$r = 330527984395124299475957654016385519914202341482140609642324395022880711289249191050673258457777458014096366590617731358671$

Polynomial Basis $t^{409} + t^{87} + 1$

$G_x =$

		060f05f	658f49c1	ad3ab189
0f718421	0efd0987	e307c84c	27accfb8	f9f67cc2
c460189e	b5aaaa62	ee222eb1	b35540cf	e9023746

$G_y =$

		1e36905	0b7c4e42	acba1dac
bf04299c	3460782f	918ea427	e6325165	e9ea10e3
da5f6c42	e9c55215	aa9ca27a	5863ec48	d8e0286b

Type 4 *Normal Basis*

$G_x =$

		1b559c7	cba2422e	3affe133
43e808b5	5e012d72	6ca0b7e6	a63aeafb	c1e3a98e
10ca0fcf	98350c3b	7f89a975	4a8e1dc0	713cec4a

$G_y =$

		16d8c42	052f07e7	713e7490
eff318ba	1abd6fef	8a5433c8	94b24f5c	817aeb79
852496fb	ee803a47	bc8a2038	78ebf1c4	99afd7d6

Curve K-571

$a = 0$

$r = 193226876150862917234767594546599367214946366485321749932861762572575957114478021226813397852270671183470671280082535146127367497406661731192968242161709250355573368527667 3$

Polynomial Basis $t^{571} + t^{10} + t^5 + t^2 + 1$

$G_x =$ 26eb7a8 59923fbc 82189631
f8103fe4 ac9ca297 0012d5d4 60248048 01841ca4
43709584 93b205e6 47da304d b4ceb08c bbd1ba39
494776fb 988b4717 4dca88c7 e2945283 a01c8972

$G_y =$ 349dc80 7f4fbf37 4f4aeade
3bca9531 4dd58cec 9f307a54 ffc61efc 006d8a2c
9d4979c0 ac44aea7 4fbebbb9 f772aedc b620b01a
7ba7af1b 320430c8 591984f6 01cd4c14 3ef1c7a3

Type 10 *Normal Basis*

$G_x =$ 04bb2db a418d0db 107adae0
03427e5d 7cc139ac b465e593 4f0bea2a b2f3622b
c29b3d5b 9aa7a1fd fd5d8be6 6057c100 8e71e484
bcd98f22 bf847642 37673674 29ef2ec5 bc3ebcf7

$G_y =$ 44cbb57 de20788d 2c952d7b
56cf39bd 3e89b189 84bd124e 751ceff4 369dd8da
c6a59e6e 745df44d 8220ce22 aa2c852c fcbbef49
ebaa98bd 2483e331 80e04286 feaa2530 50caff60