On Small Characteristic Algebraic Tori in Pairing-Based Cryptography

R. Granger, D. Page and M. Stam

University of Bristol, Department of Computer Science,
Merchant Venturers Building,
Woodland Road,
Bristol, BS8 1UB, UK.
{granger,page,stam}@cs.bris.ac.uk

Abstract. The output of the Tate pairing on an elliptic curve over a finite field may be viewed as an element of an algebraic torus. Using this simple observation, we transfer techniques recently developed for torus-based cryptography to pairing-based cryptography, resulting in more efficient computations, and lower bandwidth requirements. To illustrate the efficacy of this approach, we apply the method to pairings on supersingular elliptic curves in characteristic three.

1 Introduction

The use of pairings in cryptography is now a well-studied area, with resulting applications to identity-based encryption, key-agreement and signature schemes [5, 42], tripartite Diffie-Hellman key-agreement [28], and short signatures [6], to name just a few amongst numerous others (see e.g. [12] for a recent survey).

To support these applications much research activity has focused on developing efficient and easily implementable algorithms for their deployment [2, 17, 13]. Currently the fastest algorithm for pairing computation on elliptic curves is that of Duursma and Lee [13], which applies to the class of supersingular elliptic curves in characteristic three with so-called embedding degree six, and is preferable in pairing implementations for contemporary cryptographic schemes.

One is therefore free to use the trace-based methods found in LUC [47] and XTR [31] for post-pairing arithmetic [52], resulting in the compression of pairing outputs by a factor of two and three respectively. Scott and Barreto [44] also describe the use of traces for the computation of the pairing itself, however closer inspection of their work shows that their claim is misleading. Indeed, their method is essentially a polynomial basis transformation and hence does not offer any advantages during the computation of the pairing. Moreover, for characteristic three, we demonstrate that contrary to the claims of Scott and Barreto [44], the performance of their approach is inferior to a straightforward implementation. Thus besides pairing compression, the method they advocate does not seem to offer any benefits.

Our contribution is to achieve both efficient pairing arithmetic, and also pairing compression. Our methods are based on the simple observation that the quotient group to which the natural output of the Tate pairing belongs, may be viewed as a special representation of an algebraic torus. These groups were introduced to cryptography by Rubin and Silverberg [40], who showed under certain conditions that one can represent elements of the torus via rational embeddings into affine space, providing smaller bandwidth requirements than the corresponding field-embedded representation.

Using this property and an efficient point multiplication method developed for tori [25], we are able to perform arithmetic with pairing values that is on average 30% faster than previous methods. This is useful, for example, in pairing-based protocols where one typically blinds a point by an ephemeral random value. By bilinearity, this blinding may be performed either on the curve before the pairing evaluation, or in the extension field afterwards. Given that a pairing evaluation is usually several times more costly than either a point multiplication on the curve or an exponentiation in the field, if a pairing value ever needs to be reused, it is beneficial to compute it once and for all and to perform each ephemeral blinding in the extension field.

Examples where this occurs include the the Boneh-Franklin identity-based encryption scheme [5], the identity-based signature scheme of Hess [27], and the certificate-based encryption scheme of Gentry [21].

The aforementioned compression methods can also be used during any interactive pairing-based protocol where pairing values are transmitted between parties. Such schemes include the selective-ID identity-based encryption scheme of Boneh and Boyen [3], the interactive proof of knowledge in the short group signature scheme of Boneh *et al.* [4], and various others [22, 43].

One may regard our methods as a characteristic three version of previous work on tori [40, 25] tailored for pairings. However they may also be used for pairings on any abelian variety possessing an even embedding degree, which for efficiency reasons is the case for all contemporary pairing algorithms. As such they may also be applied to supersingular binary elliptic curves, although we do not pursue this application here since pairings based on these curves possess an inferior security/efficiency trade-off [16].

The remainder of the article is organised as follows. We next give some background on the Tate pairing and algebraic tori. In §3 we develop fast arithmetic for pairing values, and in §4 we give algorithms for efficient exponentiation. In §5, we describe the field representation we use, while in §6 we detail our improvements to the Duursma-Lee algorithm. In §7, we present implementation results, and in the final section, we make some concluding remarks and present some open problems.

2 Preliminaries

In this section we briefly provide some mathematical background, and fix some notation.

2.1 The Tate Pairing

The Tate pairing on an elliptic curve is usually computed using a variant of Miller's algorithm [34]. For the special curves often used in cryptography however, it was shown independently by Barreto *et al.* [2] and Galbraith *et al.* [17] that much of the computation of the algorithm is redundant. In terms of performance, the former paper provides the better alternative, and we refer to their algorithm as the *reduced* Tate pairing, or the BKLS algorithm.

The reduced Tate pairing. Let E be an elliptic curve over a finite field $\mathbb{F}_q$, and let $\mathcal{O}_E$ denote the identity element of the associated group of rational points on $E(\mathbb{F}_q)$. For a positive integer $l \mid \#E(\mathbb{F}_q)$ coprime to q, let $\mathbb{F}_{q^k}$ be the smallest extension field of $\mathbb{F}_q$ which contains the l-th roots of unity in $\overline{\mathbb{F}}_q$. Also, let $E(\mathbb{F}_q)[l]$ denote the subgroup of $E(\mathbb{F}_q)$ of all points of order dividing l, and similarly for the degree k extension of $\mathbb{F}_q$. From an efficiency perspective, k is usually chosen to be even [2]. For a thorough treatment of the following, we refer the reader to [2] and also [17], and to [46] for an introduction to divisors. Then assuming that $l^3 \nmid \#E(\mathbb{F}_{q^k})$, the reduced Tate pairing of order l is the map

$$e_l : E(\mathbb{F}_q)[l] \times E(\mathbb{F}_{q^k})[l] \rightarrow \mathbb{F}_{q^k}^{\times}/(\mathbb{F}_{q^k}^{\times})^l,$$

given by $e_l(P,Q) = f_{P,l}(\mathcal{D})$. Here $f_{P,l}$ is a function on E whose divisor is equivalent to $l(P) - l(\mathcal{O}_E)$, $\mathcal{D}$ is a divisor equivalent to $(Q) - (\mathcal{O}_E)$, whose support is disjoint from the support of $f_{P,l}$, and $f_{P,l}(\mathcal{D}) = \prod_i f_{P,l}(P_i)^{a_i}$, where $\mathcal{D} = \sum_i a_i P_i$. It satisfies the following properties ([15]):

- For each $P \neq \mathcal{O}_E$ there exists $Q \in E(\mathbb{F}_{q^k})[l]$ such that $e_l(P,Q) \neq 1 \in \mathbb{F}_{q^k}^{\times}/(\mathbb{F}_{q^k}^{\times})^l$ *(non-degeneracy)*;
- For any integer n, $e_l([n]P,Q) = e_l(P,[n]Q) = e_l(P,Q)^n$ for all $P \in E(\mathbb{F}_q)[l]$ and $Q \in E(\mathbb{F}_{q^k})[l]$ *(bilinearity)*;
- Let $L = hl$. Then $e_l(P,Q)^{(q^k-1)/l} = e_L(P,Q)^{(q^k-1)/L}$.

When one computes $f_{P,l}(\mathcal{D})$, the value obtained belongs to the quotient group $\mathbb{F}_{q^k}^{\times}/(\mathbb{F}_{q^k}^{\times})^l$, and not $\mathbb{F}_{q^k}^{\times}$. In this quotient, for a and b in $\mathbb{F}_{q^k}^{\times}$, $a \sim b$ if and only if there exists $c \in \mathbb{F}_{q^k}^{\times}$ such that $a = bc^l$. Clearly, this is equivalent to

$$a \sim b \text{ if and only if } a^{(q^k-1)/l} = b^{(q^k-1)/l},$$

and hence one ordinarily uses this value as the canonical representative of each coset. The isomorphism between $\mathbb{F}_{q^k}^{\times}/(\mathbb{F}_{q^k}^{\times})^l$ and the elements of order l in $\mathbb{F}_{q^k}^{\times}$ given by this exponentiation makes it possible to compute $f_{P,l}(Q)$ rather than $f_{P,l}(\mathcal{D})$ [2]. It also removes the need to compute the costly denominators in Miller's algorithm.

Fig. 1. Field definitions and Curve equations

Field	Field Polynomial	Curve	Order	MOV security
$\mathbb{F}_{3^{79}}$	$t^{79}+t^{26}+2$	$Y^2=X^3-X-1$	$3^{79}+3^{40}+1$	750
$\mathbb{F}_{3^{97}}$	$t^{97}+t^{12}+2$	$Y^2=X^3-X+1$	$(3^{97}+3^{49}+1)/7$	906
$\mathbb{F}_{3^{163}}$	$t^{163}+t^{80}+2$	$Y^2=X^3-X-1$	$3^{163}+3^{82}+1$	1548
$\mathbb{F}_{3^{193}}$	$t^{193}+t^{12}+2$	$Y^2=X^3-X-1$	$3^{193}-3^{97}+1$	1830
$\mathbb{F}_{3^{239}}$	$t^{239}+t^{24}+2$	$Y^2=X^3-X-1$	$3^{239}-3^{120}+1$	2268
$\mathbb{F}_{3^{353}}$	$t^{353}+t^{142}+2$	$Y^2=X^3-X-1$	$3^{353}+3^{177}+1$	3354

The modified Tate pairing. At Asiacrypt 2003, Duursma and Lee introduced an algorithm for pairing computation on a special family of supersingular hyperelliptic curves [13]. In common with the authors of [44], for the elliptic case, which occurs only in characteristic three, we refer to the algorithm as the *modified* Tate pairing. In Figure 1, we list a sample of curves from this family, upon which we base our implementation.

The first column gives the field over which each curve is defined, and the second lists the corresponding irreducible polynomials defining the field extensions. The third lists the curve equations and the fourth gives the order of the subgroup used. The final column gives the bit-length of the smallest finite field into which the pairing value embeds, which is a degree six extension for these curves. These parameter values were generated simply by testing which prime extension degrees yielded orders for supersingular curves that are prime, or almost prime, i.e., those possessing a small cofactor.

The modified Tate pairing improves upon the reduced variant in three ways. Firstly, using the third property listed above, instead of computing the Tate pairing of order l, one uses the pairing of order q^3+1, which eliminates the need for any point additions in Miller's algorithm. Secondly, while this apparently increases the trit-length of the exponent by a factor of three, Duursma and Lee show that the divisor computed when processing three trits at a time has a very simple form, and hence no losses are incurred. Lastly, they provide a closed form expression for the pairing, thus simplifying implementations.

We give a full description of the Duursma-Lee algorithm in §6, where we also make some elementary computational improvements.

2.2 Algebraic Tori

In 1985 El-Gamal made the suggestion that Diffie-Hellman key exchange, digital signatures and El-Gamal encryption be performed in the multiplicative group of an extension of $\mathbb{F}_p$ [14], although without going into details. Recent trends in cryptographic research have shown that by exploiting the algebraic structure not available in prime fields, one can obtain compression of elements and efficient arithmetic.

Due to the observation of Pohlig and Hellman [37], one typically works in a prime order subgroup of sufficient size in the multiplicative group of the extension field. To ensure that a particular subgroup does not embed into any subfield of the extension field, it must belong to the cyclotomic subgroup [30], which conjecturally attains the discrete logarithm security of the extension field. The public key cryptosystem XTR [31] exploits compression of elements in the cyclotomic subgroup of $\mathbb{F}_{p^6}^{\times}$ by taking their trace with respect to the quadratic subfield, to obtain a compression factor of three.

Based on similar ideas, Rubin and Silverberg [40] proposed the notion of torus-based cryptography as an alternative way to obtain compression of elements in the cyclotomic subgroup of a suitable field extension, which is isomorphic to an algebraic torus (cf. Lemma 1). The public key system CEILIDH proposed in that paper is based on the torus T_6. This torus has the property that it is birationally isomorphic to two dimensional affine space, which means that its elements can be parametrised via rational functions by only two elements of the base field, rather than the six elements ordinarily required.

It was also shown in [40] that efforts to find a natural extension of the trace-based method of XTR using symmetric functions [7], can not work. It is an open conjecture whether or not T_n is 'rational' for all n, in which case one could efficiently compress elements of T_n by a factor of $n/\phi(n)$ [53, 40]. This conjecture is known to be true when n is either a prime power, or the product of two prime powers. However, for the applications that concern us here, the status of the conjecture is unlikely to have any impact, as we explain in §6.

The torus $T_n(\mathbb{F}_q)$. Let $\mathbb{F}_q$ be a finite field where q is a power of a prime, and let Φ_n be the n-th cyclotomic polynomial. We write $G_{q,n}$ for the subgroup of $\mathbb{F}_{q^n}^{\times}$ of order $\Phi_n(q)$, and let $\mathbb{A}^n(\mathbb{F}_q)$ denote the n-dimensional affine space over $\mathbb{F}_q$, i.e., the variety whose points lie in $\mathbb{F}_q^n$.

Definition 1. *Let $k = \mathbb{F}_q$ and $L = \mathbb{F}_{q^n}$. The torus T_n is the intersection of the kernels of the norm maps $N_{L/F}$, for all subfields $k \subset F \subsetneq L$:*

$$T_n(k) := \bigcap_{k \subset F \subsetneq L} \mathrm{Ker}[N_{L/F}].$$

The following lemma provides some relevant properties of T_n [40]:

Lemma 1. *1. $T_n(\mathbb{F}_q) \cong G_{q,n}$, and thus $\#T_n(\mathbb{F}_q) = \Phi_n(q)$;*
2. If $h \in T_n(\mathbb{F}_q)$ is an element of prime order not dividing n, then h does not lie in a proper subfield of $\mathbb{F}_{q^n}/\mathbb{F}_q$.

3 The Quotient Group

Throughout this section and the remainder of the paper we assume we are working in characteristic three fields with prime extension degree (though the ideas apply equally well to arbitrary finite fields) and so where relevant, all exponents are written in ternary.

Let $l \mid \#E(\mathbb{F}_q)$ and suppose we wish to compute the modified Tate pairing of order l. Then invoking the third property of §2.1, one uses the Duursma-Lee algorithm to first compute e_{q^3+1}, which is an element in the quotient group

$$\mathcal{G} = \mathbb{F}_{q^6}^{\times}/(\mathbb{F}_{q^6}^{\times})^{q^3+1}.$$

For any $a \in \mathbb{F}_{q^6}^{\times}$ we have $a^{q^3+1} \in \mathbb{F}_{q^3}^{\times}$, and so $\mathcal{G}$ simplifies to $\mathbb{F}_{q^6}^{\times}/\mathbb{F}_{q^3}^{\times}$.

Let $G_l \subset \mathbb{F}_{q^6}^{\times}$ denote the subgroup of order l, and let $e \in \mathcal{G}$. Then the two properties:

$$\gcd(l, q^3 - 1) = 1 \text{ and } e^{q^3-1} \in G_l$$

imply that $e = gh$ for some $g \in G_l$, $h \in \mathbb{F}_{q^3}^{\times}$. Hence powering e by $q^3 - 1$ gives

$$e^{q^3-1} = (gh)^{q^3-1} = g^{q^3-1},$$

which can then be used in protocols. If a particular protocol requires an exponentiation of this value by some integer $k \bmod l$, this is performed in $\mathbb{F}_{q^6}$.

In this section we give an alternative way to obtain unique representatives of $\mathcal{G}$ easily, that furthermore permits fast multiplication, and provides automatic compression by a factor of two. We then show that the natural embedding of $\mathcal{G}$ into the extension field is just a special representation of an algebraic torus, which permits further compression.

3.1 The Basic Idea

Let $\mathbb{F}_{q^6} = \mathbb{F}_{q^3}[\sigma]/(\sigma^2 + 1)$ which is the extension we use in the Duursma-Lee algorithm. Writing $e = e_0 + e_1\sigma$ and $g = g_0 + g_1\sigma$, by the above we have

$$e = gh = g_0 h + g_1 h\sigma.$$

Since the represented coset remains invariant under multiplication by elements of $\mathbb{F}_{q^3}^{\times}$, we can divide by e_1, giving

$$e' = ee_1^{-1} = e_0/e_1 + \sigma = g_0/g_1 + \sigma.$$

This also eliminates h and may equally well be used as a canonical representative of the coset to which e belongs.

This element of the quotient group can be represented simply by the $\mathbb{F}_{q^3}$ element e_0/e_1, and thus compresses the coset representation by a factor of two. Computationally, this involves a division in $\mathbb{F}_{q^3}$.

Comparing this to powering by $q^3 - 1$, the saving is not significant, since:

$$e^{q^3-1} = \frac{e_0 - e_1\sigma}{e_0 + e_1\sigma},$$

and hence requires only a division in $\mathbb{F}_{q^6}$.

However if one exponentiates this value by some integer $k \bmod l$, this operation will be faster than if one had first powered e by $q^3 - 1$, since multiplying a generic element of $\mathcal{G}$ by this element is cheaper than multiplying two generic elements. To see this let $g = g_0/g_1 = e_0/e_1$ and $a_0 + a_1\sigma \in \mathcal{G}$. Then

$$(g + \sigma)(a_0 + a_1\sigma) = (ga_0 - a_1) + (ga_1 + a_0)\sigma,$$

which costs just two $\mathbb{F}_{q^3}$ multiplications, and not the three required if both elements are generic, in which case the arithmetic is identical to that of $\mathbb{F}_{q^6}$. If one assumes cubings and additions are essentially free, then this method will always be roughly one third faster, for whatever practical method one uses to exponentiate. The defining property of the quotient group $\mathcal{G}$ thus reduces the cost of arithmetic performed on pairing values.

3.2 Arithmetic in $\mathcal{G}$

We first introduce some terminology to clarify the operations available in $\mathcal{G}$. The property that a given coset is invariant under multiplication by elements of $\mathbb{F}_{q^3}^{\times}$ is suggestive of the projective line

$$\mathbb{P}^1(\mathbb{F}_{q^3}) = \{(x, y)/\sim\, \in (\mathbb{F}_{q^3})^2 \setminus \{(0, 0)\}\}$$

where $(x_1, y_1) \sim (x_2, y_2)$ if and only if a $\lambda \in \mathbb{F}_{q^3}^{\times}$ exists such that $(x_1, y_1) = (\lambda x_2, \lambda y_2)$. The reduction of e to e_0/e_1 may also be viewed as a map to the affine line $\mathbb{A}^1(\mathbb{F}_{q^3})$. With this analogy we introduce the following.

Definition 2. *$\mathcal{G}_P$ is the projective line $\mathbb{P}^1(\mathbb{F}_{q^3})$ endowed with the group operation induced by the arithmetic of the quadratic extension $\mathbb{F}_{q^6} = \mathbb{F}_{q^3}[\sigma]/(\sigma^2 + 1)$ via the map $(x, y) \to x + y\sigma$. The identity element is represented by the points $(\lambda, 0)$ for any $\lambda \in \mathbb{F}_{q^3}^{\times}$.*

$\mathcal{G}_A$ is the affine part of the line $\mathcal{G}_P$. The affine point corresponding to (x,y) is $X = A(x, y) = (x/y)$. Via this map the identity element is the point at infinity which we denote by $\mathcal{O}_{\mathcal{G}}$.

With this terminology it should be clear that we can mimic mixed addition methods for point multiplication on elliptic curves [9]. The use of signed digit representations follows since inverses are cheap as we show below. In §4 we derive an exponentiation algorithm using a split exponent method.

Let $P = (x, y) \in \mathcal{G}_P$ with corresponding affine representation $(X) \in \mathcal{G}_A$. We refer to the generator of $\mathrm{Gal}(\mathbb{F}_{q^6}/\mathbb{F}_q)$ as the q-Frobenius, i.e., the automorphism given by powering by q. As already stated computing the inverse of an element is virtually free. This follows since the order of $\mathcal{G}$ is $|\mathbb{F}_{q^6}^{\times}/\mathbb{F}_{q^3}^{\times}| = (q^6 - 1)/(q^3 - 1) = q^3 + 1$, and so applying the cube of the Frobenius gives the inverse: $P^{-1} = (x, -y)$ or $(-X)$ in affine. Cubing is also straightforward since we are working in characteristic three: $P^3 = (x^3, -y^3)$.

For multiplication of two points $P_1 = (x_1, y_1), P_2 = (x_2, y_2) \in \mathcal{G}_P$ with affine representations $(X_1), (X_2) \in \mathcal{G}_A$, we use the following easy lemma.

Lemma 2. *Let M and I represent the cost of a multiplication and inversion respectively in $\mathbb{F}_{q^3}$. Then the group operation for combinations of point representations is computed as follows:*

P_1	P_2	$P_1 \cdot P_2$	*Formula*	*Cost*
$\mathcal{G}_A$	$\mathcal{G}_A$	$\mathcal{G}_A$	$(X_1X_2 - 1)/(X_1 + X_2)$	$2M + I$
$\mathcal{G}_A$	$\mathcal{G}_A$	$\mathcal{G}_P$	$(X_1X_2 - 1, X_1 + X_2)$	$1M$
$\mathcal{G}_P$	$\mathcal{G}_P$	$\mathcal{G}_A$	$(x_1x_2 - y_1y_2)/(x_1y_2 + x_2y_1)$	$4M + I$
$\mathcal{G}_P$	$\mathcal{G}_P$	$\mathcal{G}_P$	$(x_1x_2 - y_1y_2, x_1y_2 + x_2y_1)$	$3M$
$\mathcal{G}_A$	$\mathcal{G}_P$	$\mathcal{G}_A$	$(X_1x_2 - y_2)/(X_1y_2 + x_2)$	$3M + I$
$\mathcal{G}_A$	$\mathcal{G}_P$	$\mathcal{G}_P$	$(X_1x_2 - y_2, X_1y_2 + x_2)$	$2M$

Squaring can naturally be performed with slightly fewer $\mathbb{F}_{q^3}$ muliplications than above; the corresponding formulae are easily deduced. Besides the precomputation necessary for the exponentiation algorithms we present in Section 4 however, squarings are not required.

With regard to exponentiations, it is clear that the mixed multiplication shown in the final row is the most efficient. If we want to compute P^k for some $k \bmod l$, we first convert P to affine and for each non-zero trit in the expansion of k perform a mixed multiplication of this point with the projective representation of the intermediate value. A multiplication with both points in projective form is equivalent to an ordinary multiplication in $\mathbb{F}_{q^6}$, so the mixed multiplication is essentially what allows the savings over arithmetic in $\mathbb{F}_{q^6}$. We exploit these observations in the exponentiation algorithms developed in §4.

3.3 An Equivalent Representation of the Quotient Group

The arithmetic just described for the quotient group is essentially identical to that developed for the torus T_2 [25, 40]. Indeed it is not difficult to see that $\mathcal{G} = T_2(\mathbb{F}_{q^3})$.

Given $e \in \mathcal{G}$, it is possible to compute the embedding e^{q^3}/e of e into $\mathbb{F}_{q^6}$ and maintain invariance under multiplication by elements of $\mathbb{F}_{q^3}^\times$. This may seem odd since $\mathbb{F}_{q^6}$ does not possess this property. However our choice of representation of elements in the subgroup of order $q^3 + 1$ makes this possible. Again let $e = e_0 + e_1\sigma$. Then

$$e^{q^3} = e_0 - e_1\sigma,$$

and hence

$$e^{q^3-1} = \frac{e_0 - e_1\sigma}{e_0 + e_1\sigma} \in G_l \subset \mathbb{F}_{q^6}. \quad (1)$$

One can perform this division in $\mathbb{F}_{q^6}$ and use the ordinary polynomial representation. Here we choose to leave this fraction unevaluated. Note that multipliying the numerator and denominator of (1) by any element of $\mathbb{F}_{q^3}^\times$ leaves the represented element unchanged.

An interesting property of this representation is that when multiplying two fractions of this form, the coefficients of the numerator and the denominator correspond exactly: let

$$c = \frac{c_0 - c_1\sigma}{c_0 + c_1\sigma}, d = \frac{d_0 - d_1\sigma}{d_0 + d_1\sigma},$$

with $c_i, d_i \in \mathbb{F}_{q^3}$. Then since $\sigma^2 + 1 = 0$, we see that

$$cd = \frac{(c_0d_0 - c_1d_1) - (c_0d_1 + c_1d_0)\sigma}{(c_0d_0 - c_1d_1) + (c_0d_1 + c_1d_0)\sigma}.$$

This also follows trivially from the fact that $(cd)^{q^3-1} = c^{q^3-1} \cdot d^{q^3-1}$.

For an implementation this allows one therefore to work with the denominator only, since one knows that the coefficients of the numerator will be identical. Hence one may view our previous operations in $\mathcal{G}$ without powering equivalently as operating purely on the denominator of (1) after powering, and so all the arithmetic carries over unchanged.

Remark 1. The representation (1) and its identification with $\mathbb{P}^1$ were given explicitly by Rubin and Silverberg [40]. However the arithmetical consequences of embedding this representation into the extension field were only fully considered in [25], where it was also noted that one may also represent T_2 as a quotient group. Thus while these ideas are not new (the representation (1) is a simple application of Hilbert's Theorem 90), they find a novel application in pairing-based cryptography.

3.4 Further Compression Using $T_6(\mathbb{F}_q)$

Since the characteristic three supersingular elliptic curves we consider have embedding degree six, one may ask why we use the arithmetic of $T_2(\mathbb{F}_{q^3})$ when the order l subgroup is in fact in $T_6(\mathbb{F}_q)$? The reason is that there seems no obvious way to utilise the extra structure provided by $T_6(\mathbb{F}_q)$ [25], though we do not rule out such a possibility. However we know that $|T_6(\mathbb{F}_q)| = (q^2 - q + 1) \mid (q^3 + 1) = |T_2(\mathbb{F}_{q^3})|$, and so $T_6(\mathbb{F}_q) \subset T_2(\mathbb{F}_{q^3})$. Thus one can use the properties of the latter and apply them to the former, utilising the improvements derived over the extension field representation.

While arithmetic improvements do not seem available with T_6, one can exploit it for better compression. As T_6 is rational, one can map nearly all its elements to the affine plane and use this representation instead for data transmissions.

Using the method described by Rubin and Silverberg [40], and thanks to some serendipitous equations for characteristic three and the given field representation, one obtains this additional compression for free.

By Definition 1,

$$T_6(\mathbb{F}_q) = \mathrm{Ker}(N_{\mathbb{F}_{q^6}/\mathbb{F}_{q^3}}) \cap \mathrm{Ker}(N_{\mathbb{F}_{q^6}/\mathbb{F}_{q^2}}) = T_2(\mathbb{F}_{q^3}) \cap \mathrm{Ker}(N_{\mathbb{F}_{q^6}/\mathbb{F}_{q^2}}).$$

To obtain a suitable representation one therefore only needs to parametrise those elements of the form (1) which have norm equal to one in the second factor.

Let $e = (a - \sigma)/(a + \sigma)$ be the compressed representation for e, and let $a = a_0 + a_1\rho + a_2\rho^2$ where $\rho^3 - \rho \pm 1 = 0$ defines the cubic extension we later use for the Duursma-Lee algorithm. Then we obtain an equation in a_0, a_1, and a_2 by the condition

$$\left(\frac{a - \sigma}{a + \sigma}\right)^{1+q^2+q^4} = 1.$$

This is equivalent to $1 + a_1^2 - a_0a_2 - a_2^2 = 0$, which one can parametrise easily with just a_1 and a_2, since $a_0 = (1 + a_1^2 - a_2^2)/a_2$. It is therefore sufficient to specify only a_1 and a_2, to describe all points on $T_6(\mathbb{F}_q)$ bar the identity, and this is essentially all that we need. We therefore have a map $\psi : \mathbb{A}^2(\mathbb{F}_q) \setminus \{(a_1, 0)\} \rightarrow T_6(\mathbb{F}_q) \setminus \{1\}$ given by

$$\psi(a_1, a_2) = \frac{((1 + a_1^2 - a_2^2) + a_1a_2\rho + a_2^2\rho^2) - a_2\sigma}{((1 + a_1^2 - a_2^2) + a_1a_2\rho + a_2^2\rho^2) + a_2\sigma}.$$

The inverse map $\psi^{-1} : T_6(\mathbb{F}_q) \setminus \{1\} \rightarrow \mathbb{A}^2(\mathbb{F}_q) \setminus \{(a_1, 0)\}$ is given as above, i.e., we just take the second and third coefficients in the fractional expression for e.

Note that $\mathbb{A}^2(\mathbb{F}_q) \setminus \{(a_1, 0)\}$ and $T_6(\mathbb{F}_q) \setminus \{1\}$ both have cardinality $q^2 - q$. In terms of the quotient group $\mathcal{G}$ and an actual pairing computation, once e_0/e_1 is computed one can use the second and third coefficients to parametrise the element, without any further computation.

Remark 2. In the context of compression, Rubin and Silverberg [39, 41] have shown how one can compress BLS short-signatures [6] by using the trace-zero subvariety contained in the Weil restriction of scalars of an elliptic curve defined over a composite field extension. This method provides a compression factor of $n/\phi(n)$ also, where $\gcd(n, 2) = 1$, and can be applied to any pairing-based protocol where one is required to transmit a point on the curve, such as [28]. However for $n \geq 5$, building upon an idea of Semaev [45], Gaudry has shown that such curves are weaker than those defined over prime field [20]. Hence this method should be regarded with some caution. We point out that this form of pre-compression is distinct from the post-compression described here, and thus these attacks do not apply.

4 Exponentiation

Now that we have set up the basic arithmetic for $\mathcal{G}$, we explore how one can optimise the basic operation of exponentiation in practice. For comparison, we also describe fast algorithms for exponentation in G_l using techniques from [49], and point multiplication in $E(\mathbb{F}_q)$, incorporating a novel technique we develop here.

For ease of notation we write the group operation for all three groups multiplicatively, and for each of the above we compare four exponentiation methods, which we detail in turn. The input to each algorithm is a base e and an integer $k \bmod l$ in standard ternary format. The output is e^k. When applicable, precomputed values are stored in affine to facilitate the mixed multiplication. We note that in all three groups inversions are essentially for free, so we consider signed digit representations.

Method 1: Signed Ternary Expansion

Using the generalised non-adjacent form, or G-NAF [8], one can take the ternary expansion of an exponent $k \bmod l$ and transform it into an equivalent signed ternary representation. Such a representation is easy to compute and reduces the average density of non-zero trits from two thirds to one half. The precomputation involves just a single squaring of the base.

Method 2: Signed Nonary Expansion

This is the same as Method 1 except we use a base nine expansion of k. This essentially halves the trit-length of k for the cost of precomputing $e^i, i = 1, ..., 8$. Again using the G-NAF, the average density of non-zero 'nits' in this expansion is four fifths.

Method 3: Sliding Window Ternary Expansion

We use an unsigned ternary expansion of k with a sliding window of width three [33][Chapter 14, Algorithm 14.85]. To do so one needs to precompute and store e^i for $0 < i < 27$ and $i \neq 0 \bmod 3$.

Method 4: Frobenius Expansion

For $e \in \mathcal{G}$ the q-Frobenius map is easily computed. Moreover, the q-th power of a compressed element is itself compressed. Since the Frobenius map satisfies $q^2 - q + 1 = 0$ (as maps) and the group order divides $q^2 - q + 1$, one can split the exponent k in two halves k_1 and k_2 where k_1, k_2 are approximately half the trit-length of l and satisfy $k \equiv k_1 + k_2 q \mod l$ [49]. One can find k_1 and k_2 very quickly having performed a one-time Gaussian two dimensional lattice basis reduction.

Thus a single exponentiation can be transformed into a double exponentiation for half the trit-length of k, for the cost of performing a double exponentiation instead. To compute e^k for a random $k \bmod l$, we perform the double exponentiation $e^{k_1}(e^q)^{k_2}$ using Shamir's trick, originally due to Straus [51]. We detail the required precomputation in the next section.

For each of k_1, k_2 we invoke the G-NAF. The average density of non-zero trits in each of their ternary expansions is $1/2$ and hence the average number of non-zero trits in the paired ternary expansion of k_1, k_2 is $1 - (1/2)^2 = 3/4$. We therefore expect to perform on average $(3/4) \cdot m/2 = (3/8)m$ multiplications of mixed type during an exponentiation.

Interestingly enough, this method also works for the elliptic curve. Clearly, one can use the same expansion of k on $E(\mathbb{F}_q)$, with powering by q is replaced by scalar multiplication by q. Somewhat surprisingly, on the curve also, multiplication by q is an efficiently computable automorphism since $[q]P = (x - (m \bmod 3)b, -y)$ for $P = (x, y)$ on the curve (where the curve equation is $Y^2 = X^3 - X + b$). Thus we arrive at a novel application of the Gallant-Lambert-Vanstone exponent split method using fast automorphisms [18].

We note that for supersingular curves over characteristic three there is also an efficient scalar multiplication algorithm due to Koblitz [29] based on the curve automorphism mapping the point (x, y) to (x^3, y^3).

4.1 Precomputation

The necessary precomputation for Methods 1, 2 and 3 is straightforward. For Method 4 we can take advantage of the q-Frobenius to reduce the cost. We use the notation of $\mathcal{G}$. Let $e = e_0 + e_1\sigma$. In order to use Shamir's trick, we need to know the values

$$(e_0/e_1 + \sigma)^{i+qj} \quad i, j \in \{0, \pm 1, \pm 2\} \tag{2}$$

in affine. Let (i, j) represent the corresponding term in (2). Then we can use the fact that for any $e \in \mathcal{G}$, we have $e^{q^2-q+1} = \mathcal{O}_{\mathcal{G}}$ to generate most of the required terms easily. To achieve this, one applies the q-Frobenius iteratively to obtain $(i, j)^q = (-j, i + j)$. We list these operations in Algorithm 1. In G_l we use the same method, having first powered e by $q^3 - 1$, but clearly without needing to obtain affine representatives.

4.2 Comparison with Trace-Based Exponentiation

The cost of a mixed multiplication in $\mathcal{G}$ is $12M$. Since $l \approx 3^m$, an exponentiation using Method 4 costs on average about $4.5mM$. This improves considerably on the $12mM$ required by the trace method of [44]. Even without mixed multiplication, this exponentiation still only requires $6.75mM$, and with neither the exponent splitting nor the mixed multiplication, this cost is only about $9mM$. Hence ordinary field arithmetic outperforms the proposed trace method, which in fact can be reduced further to about $10.3mM$ using a Euclidean algorithm [50], but is still over twice as slow.

4.3 Application to Other Pairings

We have focused primarily on small characteristic tori because the Duursma-Lee algorithm is currently the most efficient for pairing computation. In the future, the preferred embedding degree of a curve will increase in order to maintain a good security/efficiency trade-off, and thus it is likely that non-supersingular curves over large characteristic fields will be used.

Since the embedding degree n of a pairing on a given abelian variety is minimal, the output of any pairing may be considered an element of the torus T_n. Hence all of the techniques developed for torus-based cryptography may be applied, certainly for any embedding degree less than thirty.

Algorithm 1: Online Pre-computation for Double Exponentiation

input : $e = e_0 + e_1\sigma \in \mathcal{G}$

output : Representatives in $\mathcal{G}_A$ of

$$(i,j) := (e_0 + e_1\sigma)^{i+jq}, \quad i,j \in \{0, \pm 1, \pm 2\}$$

$(1,0) \leftarrow A(e)$
$(0,1) \leftarrow -(1,0)^q$
$(-1,1) \leftarrow -(0,1)^q$
$(-1,0) \leftarrow -(-1,1)^q$
$(0,-1) \leftarrow -(-1,0)^q$
$(1,-1) \leftarrow -(0,-1)^q$

$(2,0) \leftarrow \mathrm{mul}((1,0),(1,0))$
$(2,0) \leftarrow A((2,0))$
$(0,2) \leftarrow -(2,0)^q$
$(-2,2) \leftarrow -(0,2)^q$
$(-2,0) \leftarrow -(-2,2)^q$
$(0,-2) \leftarrow -(-2,0)^q$
$(2,-2) \leftarrow -(0,-2)^q$

$(1,1) \leftarrow \mathrm{mul}((1,0),(0,1))$
$(1,1) \leftarrow A((1,1))$
$(-1,2) \leftarrow -(1,1)^q$
$(-2,1) \leftarrow -(-1,2)^q$
$(-1,-1) \leftarrow -(-2,1)^q$
$(1,-2) \leftarrow -(-1,-1)^q$
$(2,-1) \leftarrow -(1,-2)^q$

$(1,2) \leftarrow \mathrm{mul}((1,0),(0,2))$
$(1,2) \leftarrow A((1,2))$
$(-1,-2) \leftarrow -(1,2)$

$(2,1) \leftarrow \mathrm{mul}((2,0),(0,1))$
$(2,1) \leftarrow A((2,1))$
$(-2,-1) \leftarrow -(2,1)$

$(2,2) \leftarrow \mathrm{mul}((2,0),(0,2))$
$(2,2) \leftarrow A((2,2))$
$(-2,-2) \leftarrow -(2,2)$

However, earlier work [25] shows that for large characteristic, the trace-based methods such as LUC [47] (for degree two extensions), and XTR [31, 50] (for degree six extensions), are slightly faster than the torus approach. For the near future however, our methods are likely to remain near-optimal.

5 Field Representation

We briefly describe efficient arithmetic for $\mathbb{F}_q$ and the required extensions.

Field Arithmetic in $\mathbb{F}_q$

Let $\mathbb{F}_q = \mathbb{F}_{3^m}$. Let $a = a_{m-1}x^{m-1} + \cdots + a_1 x + a_0$ be an element of $\mathbb{F}_q$, held in a polynomial basis, so that $a_i \in \mathbb{F}_3$. We follow other work [17, 26] and represent the element a as two bit-vectors a_H and a_L. If we let $a_H[i]$ and $a_L[i]$ denote bit i of a_H and a_L respectively, the vectors a_H and a_L are constructed from a such that for all i

$$\begin{aligned} a_H[i] &= a_i \text{ div } 2 \\ a_L[i] &= a_i \bmod 2. \end{aligned}$$

That is, a_H and a_L are a bit-sliced representation of the coefficients of a where a_H holds the high bit and a_L the low bit of a given coefficient. Given a representation of this type, we can perform a component-wise addition $r_i = a_i + b_i$ of two elements a and b using the following word-wise logical operations

$$\begin{aligned} r_H[i] &= (a_L[i] \vee b_L[i]) \oplus t \\ r_L[i] &= (a_H[i] \vee b_H[i]) \oplus t \end{aligned}$$

where

$$t = (a_L[i] \vee b_H[i]) \oplus (a_H[i] \vee b_L[i]).$$

Subtraction, and hence multiplication by two, are equally efficient since the negation of an element a simply swaps the vectors a_H and a_L over and can therefore be implemented by the same function as addition.

On a given computer with word-size w, we hold the bit-vectors a_H and a_L that represent a as two word-vectors of length $n = \lceil m/w \rceil$ and hence apply logical operations in parallel to w coefficients at a time. However, since our representation remains bit-oriented we can borrow further techniques developed for fields of characteristic two. Specifically, it is possible to construct multiplication using a variation of the often cited comb method [32] and inversion by altering the binary extended Euclidean algorithm. We used a Karatsuba method to aggressively split the multiplication operands into word sized chunks, an option that provided significant performance improvements. Unlike elements in characteristic two, squaring in characteristic three is only marginally less expensive than general multiplication. However, cubing can be performed very quickly using table-lookup in an analogous way to the so called *coefficient thinning* method in characteristic two.

Field Arithmetic in $\mathbb{F}_{q^3}$

Let $\mathbb{F}_{q^3} = \mathbb{F}_q[\rho]/(\rho^3 - \rho - b)$, with $b = \pm 1$ depending on the curve equation. Let $a = a_0 + a_1\rho + a_2\rho^2$ and $b = b_0 + b_1\rho + b_2\rho^2$ be two generic elements. We require the following operations.

***q*-Frobenius:** Since $\rho^3 = \rho + b$ we have $\rho^{3^m} = \rho + (m \bmod 3)b$ and $(\rho^2)^{3^m} = (\rho^{3^m})^2 = \rho^2 + 2b(m \bmod 3)\rho + (m^2 \bmod 3)$. Hence $a^{3^m} = (a_0 + a_1\rho + a_2\rho^2)^{3^m} = (a_0 + a_1 b(m \bmod 3) + a_2 b) + (a_1 - a_2 b(m \bmod 3))\rho + a_2\rho^2$.

Multiplication: Let $t_{00} = a_0b_0$, $t_{11} = a_1b_1$, $t_{22} = a_2b_2$, $t_{01} = (a_0 + a_1)(b_0 + b_1)$, $t_{12} = (a_1 + a_2)(b_1 + b_2)$, and $t_{20} = (a_2 + a_0)(b_2 + b_0)$. Then $ab = (t_{00} + (t_{12} - t_{11} - t_{22})b) + (t_{01} - t_{00} + t_{11} + t_{12} + t_{22}(b - 1))\rho + (t_{20} - t_{00} + t_{11})\rho^2$.

Cubing: This is straightforward in characteristic three. Since $a^3 = (a_0^3 + a_2^3 + a_1^3 b) + (a_1^3 - a_2^3 b)\rho + a_2^3\rho^2$.

Inversion: Since the extension degree is small, we can perform this directly. Let $t_{00} = a_0^2$, $t_{11} = a_1^2$, $t_{22} = a_2^2$, $t_{01} = a_0a_1$, $t_{12} = a_1a_2$, $t_{20} = a_2a_0$, and let $\Delta = a_0^3 + a_1^3 b + a_2^3 + t_{20}(a_2 - a_0) - a_1(t_{01} + t_{22}b)$. Then $a^{-1} = \Delta^{-1}((t_{00} - t_{20} + t_{22} - t_{11} - t_{12}b) + (t_{22}b - t_{01})\rho + (t_{11} - t_{20} - t_{22})\rho^2)$.

Field Arithmetic in $\mathbb{F}_{q^6}$

Let $\mathbb{F}_{q^6} = \mathbb{F}_{q^3}[\sigma]/(\sigma^2 + 1)$. Let $c = c_0 + c_1\sigma$ and $d = d_0 + d_1\sigma$ with $c_i, d_i \in \mathbb{F}_{q^3}$ be two generic elements. The arithmetic is as follows.

***q*-Frobenius:** Since $\sigma^2 = -1$, we have that $\sigma^3 = -\sigma$ and as m is odd, we obtain $c^{3^m} = c_0^{3^m} - c_1^{3^m}\sigma$.

Multiplication: Let $t_{00} = c_0d_0$, $t_{11} = c_1d_1$, and $t_{01} = (c_0 + c_1)(d_0 + d_1)$. Then $cd = (t_{00} - t_{11}) + (t_{01} - t_{00} - t_{11})\sigma$.

Cubing: $c^3 = c_0^3 - c_1^3\sigma$.

Inversion: Let $\Delta = c_0^2 + c_1^2$. Then $c^{-1} = \Delta^{-1}(c_0 - c_1\sigma)$.

6 The Modified Tate Pairing Algorithm

In this section we detail how to efficiently implement the Duursma-Lee algorithm for the computation of the modified Tate pairing.

Let $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ be points of order l. Then the modified Tate pairing on the supersingular curve $E(\mathbb{F}_q) : Y^2 = X^3 - X + b$ is the mapping $f_P(\phi(Q))^{q^3-1}$ where $\phi : E(\mathbb{F}_q) \to E(\mathbb{F}_{q^6})$ is the distortion map $\phi(x_2, y_2) = (\rho - x_2, \sigma y_2)$. However, making use of the techniques of §3 we do not need to perform the final powering, as we presume the output will be stored and transmitted in compressed form (Algorithm 2) .

Algorithm 2: The *Duursma-Lee* Algorithm

input : point $P = (x_1, y_1)$, point $Q = (x_2, y_2)$
output : $f_P(\phi(Q)) \in \mathcal{G}$

$f \leftarrow 1$
for $i = 1$ **to** m **do**
  $x_1 \leftarrow x_1^3, y_1 \leftarrow y_1^3$
  $\mu \leftarrow x_1 + x_2 + b, \lambda \leftarrow -y_1 y_2 \sigma - \mu^2$
  $g \leftarrow \lambda - \mu\rho - \rho^2, f \leftarrow f \cdot g$
  $x_2 \leftarrow x_2^{1/3}, y_2 \leftarrow y_2^{1/3}$
end
return f

6.1 Cost Analysis

Let M denote the cost of an $\mathbb{F}_q$ multiplication. Each iteration of the loop requires $2M$ to compute μ^2 and $y_1 y_2$, and an $\mathbb{F}_{q^6}$ multiplication to compute $f \cdot g$. Since a generic $\mathbb{F}_{q^6}$ multiplication costs $18M$, Scott and Barreto [44] reckon that besides the necessary cubings and cube roots, each loop iteration costs $20M$. However, in each iteration g is sparse, i.e., not all of its terms are non-trivial, One can exploit this to reduce the cost of multiplying g and f, which is not sparse in general, to $13M$. This total of $15M$ improves on the trace-based method suggested by Scott and Barreto. In fact one can reduce the cost for each loop iteration in the ordinary Duursma-Lee algorithm to just $14M$, by unrolling the main loop and better exploiting the sparsity of g.

Algorithm 3: A Refined *Duursma-Lee* Algorithm.

input : point $P = (x_1, y_1)$, point $Q = (x_2, y_2)$
output : $f_P(\phi(Q)) \in \mathcal{G}$

$f \leftarrow 1$
for $i = 1$ **to** $(m-1)/2$ **do**
  $x_1 \leftarrow x_1^3, y_1 \leftarrow y_1^3$
  $\mu \leftarrow x_1 + x_2 + b, \lambda \leftarrow -y_1 y_2 \sigma - \mu^2$
  $g_1 \leftarrow \lambda - \mu\rho - \rho^2$
  $x_2 \leftarrow x_2^{1/3}, y_2 \leftarrow y_2^{1/3}$
  $x_1 \leftarrow x_1^3, y_1 \leftarrow y_1^3$
  $\mu \leftarrow x_1 + x_2 + b, \lambda \leftarrow -y_1 y_2 \sigma - \mu^2$
  $g_2 \leftarrow \lambda - \mu\rho - \rho^2$
  $g \leftarrow g_1 g_2, f \leftarrow f \cdot g$
  $x_2 \leftarrow x_2^{1/3}, y_2 \leftarrow y_2^{1/3}$
end
$x_1 \leftarrow x_1^3, y_1 \leftarrow y_1^3$
$\mu \leftarrow x_1 + x_2 + b, \lambda \leftarrow -y_1 y_2 \sigma - \mu^2$
$g \leftarrow \lambda - \mu\rho - \rho^2, f \leftarrow f \cdot g$
return f

We demonstrate this technique in Algorithm 3 which provides a saving since in each loop, multiplying g_1 by g_2 costs only $6M$. Multiplying g by f in each loop costs $18M$ since they are both generic $\mathbb{F}_{q^6}$ elements. Both μ^2 and $y_1 y_2$ are computed twice in each loop: once for g_1 and once for g_2. In total the cost therefore is $(6M + 4M)(m-1)/2 + 18M(m-3)/2 + 13M = 14mM - 19M$, which is equivalent to about $14M$ per loop iteration of Algorithm 2.

This cost analysis ignores the cost of computing cubings and cube roots. Because of the large number of times each of these operations are invoked, it has been suggested that one should use normal bases to accommodate them efficiently, since they are then implemented using cyclic shifts. Normal bases are well-studied in even characteristic, but for characteristic three one can not construct optimal, type one normal bases with prime extension degree [19, 36], although type two bases are available for some values of m. As a result, the cost of general multiplication in software is relatively large, even when variations of high performance methods in characteristic two are used [38, 35]. For example, we found that when $m = 239$ normal basis multiplication is between two and three times slower than a polynomial basis multiplication. However, in hardware implementations on a smart-card for example, normal bases still seem the obvious choice since they can match the multiplication speed of polynomial basis while offering inexpensive cube and cube root operations, although perhaps at the cost of flexibility.

To reduce the cost of computing cube roots using a polynomial basis, we observe that the successive cube roots of x_2 and y_2 can be computed more easily in reverse order and stored for the duration of the algorithm. Since for any $x_2 \in \mathbb{F}_q$, we have $x_2 = x_2^{3^m}$, the required values $x_2^{1/3^i}$ can be computed as $x_2^{3^{m-i}}$, and thus one does not need to compute any cube roots at all. The memory requirement for this is only about $2^{-11}m^2$ Kb and the time taken is just the cost of $2m$ cubings. If memory is at a premium, one can reduce this to about $2^{-4.5}m^{3/2}$ Kb with double the number of cubings using further loop unrolling and pebbling strategies.

Remark 3. As already mentioned, Scott and Barreto's method [44] is effectively a change of basis and not a compressed method of computing a pairing. Hence it is unsurprising that the loop unrolling strategy of Algorithm 3 can be used to reduce the cost of the trace method given there, as kindly pointed out by Barreto [1].

Remark 4. Scott and Barreto [44] suggested an open problem asking if it possible to perform the pairing computation directly in compressed form for some compression factor ≥ 3 on ordinary (non-supersingular) curves in characteristic $p > 3$. A compression factor larger than 3 is extremely unlikely. For pairing-based applications, the desirable extension degrees in the near future are likely to remain small, and no larger than twenty. By Lemma 1, the maximum compression factor possible for a given extension degree n is $n/\phi(n)$, and for $n < 20$, this maximum is three, which is already achieved for the modified Tate pairing.

Fig. 2. Pairing and Exponentiation Timings.

	$\mathbb{F}_{3^{79}}$	$\mathbb{F}_{3^{97}}$	$\mathbb{F}_{3^{163}}$	$\mathbb{F}_{3^{193}}$	$\mathbb{F}_{3^{239}}$	$\mathbb{F}_{3^{353}}$
Pairing						
BKLS	$13.96ms$	$23.60ms$	$79.11ms$	$123.21ms$	$179.30ms$	$527.56ms$
Algorithm 3	$4.67ms$	$8.41ms$	$29.26ms$	$45.67ms$	$65.73ms$	$197.58ms$
Exponentiation in G_l						
Method 1	$3.65ms$	$6.14ms$	$20.98ms$	$33.21ms$	$44.72ms$	$130.27ms$
Method 2	$4.57ms$	$7.25ms$	$21.53ms$	$31.61ms$	$43.56ms$	$119.16ms$
Method 3	$3.67ms$	$5.79ms$	$17.85ms$	$26.69ms$	$36.45ms$	$101.75ms$
Method 4	$3.06ms$	$5.10ms$	$16.55ms$	$24.67ms$	$34.74ms$	$99.56ms$
Exponentiation in $\mathcal{G}$						
Method 1	$2.55ms$	$4.27ms$	$14.15ms$	$21.67ms$	$30.69ms$	$88.06ms$
Method 2	$2.62ms$	$5.21ms$	$13.21ms$	$20.38ms$	$26.97ms$	$74.90ms$
Method 3	$3.69ms$	$4.72ms$	$15.78ms$	$22.96ms$	$37.96ms$	$73.29ms$
Method 4	$2.32ms$	$4.07ms$	$11.84ms$	$17.63ms$	$24.73ms$	$69.30ms$
Point Multiplication in $E(\mathbb{F}_q)$						
Method 1	$1.83ms$	$3.11ms$	$10.62ms$	$16.94ms$	$24.11ms$	$69.78ms$
Method 2	$1.72ms$	$2.84ms$	$9.47ms$	$14.73ms$	$21.15ms$	$60.70ms$
Method 3	$1.82ms$	$3.01ms$	$9.66ms$	$14.95ms$	$21.19ms$	$58.70ms$
Method 4	$1.18ms$	$1.95ms$	$8.11ms$	$12.75ms$	$19.04ms$	$55.93ms$

7 Implementation Results

In order to provide some concrete idea of the practical cost of our own and other methods, we implemented the proposed field arithmetic, pairing algorithms and exponentiation methods. We used a GCC 3.3 compiler suite to build our implementation and ran timing experiments on a Linux based PC incorporating a 2.80 GHz Intel Pentium 4 processor. The entire system was constructed in C++. We accept that further performance improvements could be made through aggressive profiling and optimisation but are confident our results are representative of the underlying algorithms and allow a comparison between them.

Figure 2 shows the result of timing this implementation using a variety of different base field sizes. In the pairing section, Algorithm 3 refers to the augmented version of Duursma-Lee presented in this paper, with the cube root precomputation strategy and the loop unrolling. The BKLS method is included as a reference. We do not include timings for the methods of [44] since our operation count clearly shows they will be slower than our alternatives. Figure 3 gives timings for the underlying field operations.

We note first that our implementation of Algorithm 3 is between two to three times faster than the BKLS algorithm. With regard to exponentiation, Method 4 is the most efficient for all field sizes and in all three groups, and in $\mathcal{G}$ is nearly twice as fast as Method 1 in G_l. Contrary to a claim of Koblitz [29] that the ratio of the time required for an exponentation in $\mathbb{F}_{q^6}$ to the time required for a point multiplication in $E(\mathbb{F}_q)$ is 12, our results demonstrate that for fields

Fig. 3. Timings for Field Operations.

	$\mathbb{F}_{3^{79}}$	$\mathbb{F}_{3^{97}}$	$\mathbb{F}_{3^{163}}$	$\mathbb{F}_{3^{193}}$	$\mathbb{F}_{3^{239}}$	$\mathbb{F}_{3^{353}}$
$\mathbb{F}_q$						
Add	$0.55\mu s$	$0.53\mu s$	$0.58\mu s$	$0.63\mu s$	$0.61\mu s$	$0.64\mu s$
Square	$4.42\mu s$	$6.07\mu s$	$12.99\mu s$	$16.48\mu s$	$19.48\mu s$	$40.97\mu s$
Cube	$0.85\mu s$	$0.84\mu s$	$0.96\mu s$	$1.26\mu s$	$1.24\mu s$	$1.77\mu s$
Invert	$23.18\mu s$	$33.26\mu s$	$70.10\mu s$	$97.20\mu s$	$136.86\mu s$	$303.27\mu s$
Multiply	$4.06\mu s$	$6.02\mu s$	$12.80\mu s$	$17.83\mu s$	$19.42\mu s$	$43.11\mu s$
$\mathbb{F}_{q^3}$						
Add	$0.60\mu s$	$0.60\mu s$	$0.80\mu s$	$0.90\mu s$	$0.90\mu s$	$0.50\mu s$
Cube	$2.10\mu s$	$2.10\mu s$	$2.30\mu s$	$2.50\mu s$	$3.20\mu s$	$4.20\mu s$
Invert	$65.00\mu s$	$94.70\mu s$	$204.40\mu s$	$275.90\mu s$	$350.60\mu s$	$741.80\mu s$
Frobenius	$1.10\mu s$	$0.90\mu s$	$1.10\mu s$	$1.00\mu s$	$1.30\mu s$	$1.40\mu s$
Multiply	$26.10\mu s$	$37.80\mu s$	$74.20\mu s$	$98.00\mu s$	$115.50\mu s$	$249.00\mu s$
$\mathbb{F}_{q^6}$						
Add	$0.90\mu s$	$0.90\mu s$	$0.90\mu s$	$1.10\mu s$	$1.00\mu s$	$1.10\mu s$
Cube	$2.80\mu s$	$4.60\mu s$	$4.40\mu s$	$4.00\mu s$	$5.00\mu s$	$5.60\mu s$
Invert	$165.50\mu s$	$237.20\mu s$	$497.40\mu s$	$670.10\mu s$	$817.10\mu s$	$1709.50\mu s$
Frobenius	$2.00\mu s$	$2.10\mu s$	$1.90\mu s$	$2.00\mu s$	$2.10\mu s$	$2.10\mu s$
Multiply	$75.70\mu s$	$106.10\mu s$	$227.10\mu s$	$296.80\mu s$	$347.30\mu s$	$745.10\mu s$

of a cryptographic size, this value is in fact closer to 1.3. Thus the techniques from [49], together with the fast multiplication in $\mathcal{G}$, improve the efficiency of post-pairing arithmetic considerably.

We conceed that while we have not implemented Koblitz's complex multiplication exponentiation method, due to the estimated large preprocessing time required, we do not think it would affect this comparison significantly.

Furthermore, due to our direct inversion method, the ratio of inversion time to multiplication time in $\mathbb{F}_{q^3}$ is under three for all field sizes. This means our compression method in $\mathcal{G}$ costs roughly $4/3$ multiplications in $\mathbb{F}_{q^6}$, and is therefore also very efficient.

8 Conclusion and Open Problems

We have shown how to take advantage of the quotient group to which a pairing value naturally belongs in order to speed up exponentiations, and to obtain fast compression of pairing values. We have also proposed some simple refinements to the Duursma-Lee algorithm to improve efficiency. Our results strongly indicate that there are definite advantages to implementing pairing-based cryptographic protocols in characteristic three: the often quoted value of ten for the ratio of the speed of a pairing evaluation to a point multiplication on the curve is really closer to three or four.

Some issues remain. One could certainly improve the exponentiation times for all three groups if there exists an efficiently computable ternary analogue of the

Joint Sparse Form [48]. With regard to side channel attacks, such a method may be undesirable since one can not render cubing and multiplication in charcteristic three fields indistinguishable without a serious detriment to performance. As such, a cube-and-multiply-always method using the exponent splitting of Method 4 will half the cost of a secure full length expansion.

Also the exact security of the discrete logarithm problem in characateristic three using the ternary analogue of Coppersmith's method has yet to be investigated [10, 11]. Preliminary research into this problem using Adleman's Function Field Sieve has been conducted [23, 24] but the problem should still be considered open.

Lastly, do there exist methods for faster pairing evaluation using MNT curves, and how might they compare to those presented here?

Acknowledgements

The authors would like to thank the anonymous referee, Paulo Barreto, Steven Galbraith, Keith Harrison, Karl Rubin, Mike Scott, Alice Silverberg, Nigel Smart and Fré Vercauteren for many helpful comments and fruitful discussions.

References

1. P. Barreto. Personal Communication.
2. P. Barreto, H. Kim, B. Lynn and M. Scott. Efficient Algorithms for Pairing-Based Cryptosystems. In *Advances in Cryptology (CRYPTO 2002)*, Springer LNCS 2442, 354–368, 2002.
3. D. Boneh and X. Boyen. Efficient Selective-ID Secure Identity-Based Encryption Without Random Oracles. In *Advances in Cryptology (EUROCRYPT 2004)*, Springer LNCS 3027, 223–238, 2004.
4. D. Boneh, X. Boyen and H. Shacham. Short Group Signatures. In *Advances in Cryptology (CRYPTO 2004)*, Springer LNCS 3152, 41–55, 2004.
5. D. Boneh and M. Franklin. Identity-Based Encryption from the Weil Pairing. In *SIAM Jounal on Computing*, Volume 32, no. 3, 586-615, 2003.
6. D. Boneh, B. Lynn and H. Shacham. Short signatures from the Weil Pairing. In *Advances in Cryptology (ASIACRYPT 2001)*, Springer LNCS 2248, 514–532, 2001.
7. W. Bosma, J. Hutton and E. Verheul. Looking beyond XTR. In *Advances in Cryptology (ASIACRYPT 2002)*, Springer LNCS 2501, 46–63, 2002.
8. W. Clark and J. Liang. On arithmetic weight for a general radix representation of integers. In *IEEE Trans. Info. Theory*, **19**, 823–826, 1973.
9. H. Cohen, A. Miyaji and T. Ono. Efficient elliptic curve exponentiation using mixed coordinates. In *Advances in Cryptology (ASIACRYPT 1998)*. Springer LNCS 1514 , 51–65, 1998.
10. D. Coppersmith. Evaluating logarithms in GF(2^n). In *16th ACM Symp. Theory of Computing*, 201–107, 1984.
11. D. Coppersmith. Fast evaluation of logarithms in fields of characteristic two. *IEEE Trans. Info. Theory*, **30**, 587–594, July 1984.
12. R. Dutta, R. Barua and P. Sarkar. Pairing-Based Cryptographic Protocols: A Survey. Cryptology ePrint Archive, Report 2004/064. Available from `http://eprint.iacr.org/2004/064`.

13. I. Duursma and H. Lee. Tate Pairing Implementation for Hyperelliptic Curves $y^2 = x^p - x + d$. In *Advances in Cryptology (ASIACRYPT 2003)*, Springer LNCS 2894, 111–123, 2003.
14. T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *IEEE Trans. Info. Theory* **31**, 469–472, 1985.
15. G. Frey and H. Ruck. A Remark Concerning m-Divisibility and the Discrete Logarithm Problem in the Divisor Class Group of Curves. In *Math. Comp. 62*, 865-874, 1994.
16. S. Galbraith. Supersingular Curves in Cryptography. In *Advances in Cryptology (ASIACRYPT 2001)*, Springer LNCS 2248, 495–513, 2001.
17. S. Galbraith, K. Harrison and D. Soldera. Implementing the Tate pairing. In *Proc. of ANTS V*, Springer LNCS 2369, 324–337, 2002.
18. R. Gallant, J. Lambert and S. Vanstone. Faster Point Multiplication on Elliptic Curves with Efficient Endomorphisms. In *Advances in Cryptology (CRYPTO 2001)*, Springer LNCS 2139, 190–200, 2001.
19. S. Gao. Normal Bases over Finite Fields. PhD Thesis, Waterloo University, 1993.
20. P. Gaudry Index calculus for abelian varieties and the elliptic curve discrete logarithm problem Cryptology ePrint Archive, Report 2004/073. Available from `http://eprint.iacr.org/2004/073`.
21. C. Gentry. Certificate-Based Encryption and the Certificate Revocation Problem. In *Advances in Cryptology (EUROCRYPT 2003)*, Springer LNCS 2656, 272–293, 2003.
22. P Golle and A. Juels. Dining Cryptographers Revisited. In *Advances in Cryptology (EUROCRYPT 2004)*, Springer LNCS 3027, 456–473, 2004.
23. R. Granger. Estimates for discrete logarithm computations in finite fields of small characteristic. In *Cryptography and Coding*, Springer LNCS 2898, 190–206, 2003.
24. R. Granger, A. Holt, D. Page, N. Smart and F. Vercauteren. Function Field Sieve in Characteristic Three. In *Proc. of ANTS VI*, Springer LNCS 3076, 223–234, 2004.
25. R. Granger, D. Page and M. Stam. A Comparison of CEILIDH and XTR. In *Proc. of ANTS VI*, Springer LNCS 3076, 235–249, 2004.
26. K. Harrison, D. Page and N.P. Smart. Software Implementation of Finite Fields of Characteristic Three, for use in Pairing Based Cryptosystems. In *LMS Journal of Computation and Mathematics*, **5** (1), 181–193, London Mathematical Society, 2002.
27. F. Hess. Efficient Identity based Signature Schemes based on Pairings. In *Selected Areas in Cryptography (SAC 2002)*, Springer LNCS 2595, 310–324, 2003.
28. A. Joux. A One Round Protocol for Tripartite Diffie-Hellman. In *Proc. of ANTS IV*, Springer LNCS 1838, 385–394, 2000.
29. N. Koblitz. An elliptic curve implementation of the finite field digital signature algorithm. *Advances in Cryptology (CRYPTO 98)*, Springer LNCS 1462, 327–337, 1998.
30. A. Lenstra. Using Cyclotomic Polynomials to Construct Efficient Discrete Logarithm Cryptosystems over Finite Fields. In *Proc. of ACISP97*, Springer LNCS 1270, 127–138, 1997.
31. A. Lenstra and E. Verheul. The XTR Public Key System. In *Advances in Cryptology (CRYPTO 2000)*, Springer LNCS 1880, 1–19, 2000.
32. J. López and R. Dahab. High Speed Software Multiplication in $\mathbb{F}_{2^m}$. In *Progress in Cryptography (INDOCRYPT 2000)*, Springer-Verlag LNCS 1977, 203–212, 2000.
33. A.J. Menezes, P.C. van Oorschot and S.A. Vanstone. Handbook of Applied Cryptography. CRC Press, 1997.

34. V. Miller. Short programs for functions on curves. Unpuplished manuscript, 1986. Available from `http://crypto.stanford.edu/miller/miller.pdf`.
35. P. Ning and Y.L. Yin. Efficient Software Implementation for Finite Field Multiplication in Normal Basis. In *Information and Communications Security (ICICS)*, Springer-Verlag LNCS 2229, 177–188, 2001.
36. M. Nöcker. Data structures for parallel exponentiation in finite fields. PhD Thesis, Universität Paderborn, 2001.
37. G. Pohlig and M. Hellman. An improved algorithm for computing discrete logarithms over $GF(p)$ and its cryptographic significance. In *IEEE Trans. Info. Theory* **24**, 106–110, 1978.
38. A. Reyhani-Masoleh and M.A. Hasan: Fast Normal Basis Multiplication Using General Purpose Processors. In *Selected Areas in Cryptography (SAC 2001)*, Springer LNCS 2259, 230–244, 2001.
39. K. Rubin and A. Silverberg. Supersingular abelian varieties in cryptology. In *Advances in Cryptology (CRYPTO 2002)*, Springer LNCS 2442, 336–353, 2002.
40. K. Rubin and A. Silverberg. Torus-Based Cryptography. In *Advances in Cryptology (CRYPTO 2003)*, Springer LNCS 2729, 349–365, 2003.
41. K. Rubin and A. Silverberg. Using Primitive Subgroups to Do More with Fewer Bits. In *Algorithm Number Theory (ANTS-VI)*, Springer LNCS 3076, 18–41, 2004.
42. R. Sakai, K. Ohgishi and M. Kasahara. Cryptosystems Based on Pairings. In *Symposium on Cryptography and Information Security 2000 (SCIS2000)*, Okinawa, Japan, Jan 26–28, 2000.
43. M. Scott. Authenticated ID-based Key Exchange and remote log-in with insecure token and PIN number. Cryptology ePrint Archive, Report 2002/164. Available from `http://eprint.iacr.org/2002/164`.
44. M. Scott and P. Barreto. Compressed Pairings. In *Advances in Cryptology (CRYPTO 2004)*, Springer LNCS 3152, 140–156, 2004.
45. I. Semaev. Summation polynomials and the discrete logarithm problem on elliptic curves. Cryptology ePrint Archive, Report 2004/031. Available from `http://eprint.iacr.org/2004/031`.
46. J. Silverman. The arithmetic of elliptic curves. Springer GTM 106, 1986.
47. P. Smith and C. Skinner. A public-key cryptosystem and a digital signature system based on the Lucas function analogue to discrete logarithms. In *Advances in Cryptology (ASIACRYPT 1995)*, Springer LNCS 917, 357–364, 1995.
48. J.A. Solinas. Low-Weight Binary Representations for Pairs of Integers. University of Waterloo, Technical Report CORR 2001-41.
49. M. Stam and A. Lenstra. Efficient Subgroup Exponentiation in Quadratic and Sixth Degree Extensions. In *Cryptographic Hardware and Embedded Systems (CHES 2002)*, Springer LNCS 2523, 318–332, 2002.
50. M. Stam and A. Lenstra. Speeding Up XTR. In *Advances in Cryptology (ASIACRYPT 2001)*, Springer LNCS 2248, 125–143, 2001.
51. E.G. Straus. Problems and Solutions: (5125) Addition Chains of Vectors. In *American Mathematical Monthly*, **71**, 806–808, 1964.
52. E. Verheul. Personal Communication, 2001.
53. V.E. Voskresenskiĭ. Algebraic Groups and Their Birational Invariants. In *Translations of Mathematical Monographs*, **179**, American Mathematical Society, 1998.