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 is an element in the multiplicative group of an extension field modulo a particular subgroup. One ordinarily powers this element to obtain a unique representative for the output coset, and performs any further necessary arithmetic in the extension field. Rather than an obstruction, we show to the contrary that one can exploit this quotient group to eliminate the final powering, to speed up exponentiations and to obtain a simple compression of pairing values which is useful during interactive identity-based cryptographic protocols. Specifically we demonstrate that methods available for fast point multiplication on elliptic curves such as mixed addition, signed digit representations and Frobenius expansions, all transfer easily to the quotient group, and provide a significant improvement over the arithmetic of the extension field. We also show that the natural embedding of this group into the extension field may be interpreted as a special representation of an algebraic torus, which for supersingular curves in characteristic three with MOV embedding degree six, permits a higher compression factor than is possible in the quotient group. To illustrate the efficacy of our methods, we apply them to the basic arithmetic required in pairing-based cryptography using these curves.

1 Introduction

Since finding their first application in the identity-based key agreement and signature schemes of Sakai, Ohgishi and Kasahara [42], the existence of efficiently computable, non-degenerate bilinear maps, or pairings, has allowed cryptographers to explore avenues of research which had previously been uninstantiable [46]. Originally used as a method of attack on elliptic curve cryptosystems [31, 12], the constructive potential of pairings based on elliptic curves is now well-studied. Positive uses of pairings include the tripartite Diffie-Hellman protocol of Joux [25], the identity-based encryption scheme of Boneh and Franklin [4], the short signature scheme of Boneh *et al.* [5], the identity-based signature scheme of Hess [24], and numerous others [48, 54, 37, 30]. To support these applications much research activity has focused on developing efficient and easily

implementable algorithms for their deployment [1, 14, 10]: it is to this area this article contributes.

The natural output of the Tate pairing is an element in the multiplicative group of an extension of the underlying finite field over which the elliptic curve group is defined, modulo a particular subgroup. Since for applications one requires a unique representative for the output coset, one typically exponentiates the pairing value by the index of the subgroup to eliminate any ambiguity. What remains is an element of the subgroup which can then be used in cryptographic protocols.

In many situations it would be useful to speed up exponentiations in this subgroup. For example, in pairing-based protocols one typically blinds a point by an ephemeral random value. The bilinearity property of the pairing allows the blinding to be performed either on the curve before the pairing evaluation, or in the extension field afterwards. Since a pairing computation is usually several times more expensive 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 compute it once and for all and perform each ephemeral blinding in the extension field.

For instance, in the Boneh-Franklin identity-based encryption scheme [4] should Bob ever want to send more than one message to Alice he should compute a single pairing dependent upon Alice's identity and then exponentiate by an ephemeral value in the extension field for each message. Similarly in the identity-based signature scheme of Hess [24], if Alice needs to sign more than one message at all she should compute a single pairing and thereafter needs only exponentiate in the extension field once per signature. If Bob expects to receive more than one signature from Alice then he too can save one pairing computation per verification by precomputing a pairing and exponentiating in the field. As a final example consider the certificate-based encryption scheme of Gentry [18]. Here the value being exponentiated is a product of two or more pairings, one of which can be precomputed. In this scenario it makes no sense at all to blind a point on the curve since an exponentiation must take place in the extension field to blind the precomputed pairing value.

Fortunately the quotient group to which a pairing value belongs possesses some useful properties one can exploit. Firstly we show that the defining quality of this quotient group allows one to perform a multiplication more cheaply than is possible in the extension field. This results in an exponentiation algorithm which is nearly twice as fast as a generalised non-adjacent form ternary cube and multiply method [7]. The methods we use for exponentiation are derived from those developed for fast point multiplication on general elliptic curves such as mixed addition, signed digit representations and Frobenius expansions, which all transfer easily to the quotient group.

By using the Duursma-Lee algorithm for pairing computation on supersingular curves in characteristic three with embedding degree six [10], the same defining property of the quotient group also allows one to obtain a unique coset representative for the pairing output without powering. This automatically gives

a compressed representation of that element, reducing its size by a factor of two. The powering itself can be regarded as an embedding of the quotient group into the field extension. Our approach shows that this is unnecessary since one does not need to map to the extension field at all: every protocol that uses the Tate pairing can be performed in the quotient group to which pairing values naturally belong. Therefore rather than complicating implementations, one can take advantage of the quotient group to simplify implementations, and obtain a significant performance improvement by doing so.

Furthermore we show that the natural embedding of the quotient group into the extension field is no more than a special representation of an algebraic torus. In this representation also the final powering usually required to obtain a unique coset representative is performed implicitly and in addition, all the operations available for fast exponentiation in the quotient group apply.

The reason for this is that the subgroup of the extension field into which the quotient group embeds has the property that it does not embed into any subfield of the full field extension [31, 13], and hence is a subgroup of the cyclotomic subgroup [27]. An equivalent definition of the cyclotomic subgroup is the algebraic torus [41]. An algebraic torus over a finite field $\mathbb{F}_q$ is just an algebraic group whose group law is ordinary multiplication in some extension field. Specifically, for any positive integer n one can define an algebraic torus T_n over $\mathbb{F}_q$ such that over $\mathbb{F}_{q^n}$, this variety is isomorphic to $\phi(n)$ copies of $\mathbb{F}_q^*$ where $\phi(n)$ is the dimension of T_n [55, 41]. Hence algebraic tori occur quite naturally in the context of pairings.

Scott and Barreto [44] note that one is free therefore to use the trace-based compression methods available for small-degree extensions of finite fields [49, 28]. For the main case we consider - supersingular elliptic curves in characteristic three with embedding degree six - using the algebraic torus perspective [41], we match the compression rate of [44] who use XTR [28], without any extra computation. In large characteristic the trace-based approach may offer some advantages over our methods (see [22] for a comparison between the torus and trace-based approach). However, in small characteristic, using XTR for exponentation is in fact slower than a naive implementation of $\mathbb{F}_{q^6}$ due to the low cost of cubing: our algorithms are over twice as fast.

The compression methods we derive have applications to interactive pairing-based protocols, where pairing values can be transmitted between parties and hence using these methods provides a reduced bandwidth requirement. Such schemes include the selective-ID identity-based encryption scheme of Boneh and Boyen [2], the interactive proof of knowledge in the short group signature scheme of Boneh *et al.* [3], and various others [19, 43].

Due to the aforementioned advantages over other useful groups, in this article we restrict our attention to the use of supersingular elliptic curves in characteristic three. These curves also offer some benefits over other parameter selections. The embedding degree of six, while not optimal in terms of the relative security requirements for discrete logarithm algorithms in characteristic three finite fields and elliptic curves, still offers a good security/efficiency trade-off for contempo-

rary key-size recommendations. In particular the Duursma-Lee algorithm for the Tate pairing is considerably faster than the BKLS algorithm available for even and large characteristics [10, 1]. Hence we focus on optimising this instantiation of the Tate pairing.

One may regard our methods as a characteristic three version of [22] tailored for pairings. They are developed assuming only that the embedding degree of the curve is even since in practice these curves are more efficient than the alternative [1]. As such they may also be applied to supersingular binary curves from embedding degree four with almost no modifications, although since pairings based on these curves possess an inferior security/efficiency trade-off [13], we comment on this application only briefly.

The remainder of the article is organised as follows. In the next section we give some background on algebraic tori and the Tate pairing. In Section 3 we develop fast arithmetic for the quotient group and introduce our special representation for the relevant algebraic torus. In Section 4 we give algorithms for efficient exponentiation and in Section 5, describe the field representation we use. In Section 6 we give a fast implementation of the Tate pairing, and in Section 7, give our implementation results. Lastly, we make some concluding remarks and present a number of open problems.

2 Preliminaries

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

2.1 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 $\mathbb{F}_p$ [11]. At the time there was no real application. Since then we have learnt that in doing so one can exploit the algebraic structure not available in prime fields to obtain compression of elements and efficient arithmetic.

Due to the observation of Pohlig and Hellman [38], 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 [27], which conjecturally attains the discrete logarithm security of the extension field. The public key cryptosystem XTR [28] exploits compression of elements in the cyclotomic subgroup of $\mathbb{F}_{p^6}^*$ by taking their trace with respect to the quadratic subfield, to obtain a compression factor of three.

Rubin and Silverberg [41] 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 (see Lemma 1). The public key system CEILIDH proposed in that paper uses the torus $T_6(\mathbb{F}_p)$. This torus has the property that it is birationally isomorphic

to two dimensional affine space over $\mathbb{F}_p$, which means that its elements can be represented by only two elements of $\mathbb{F}_p$ in polynomial time, rather than the six elements of $\mathbb{F}_p$ ordinarily required.

It was also shown in [41] that efforts to find a natural extension of the trace-based method of XTR using symmetric functions can not work [6]. It is an open conjecture whether or not T_n is 'rational' for all n, in which case one could compress elements of T_n by a factor of $n/\phi(n)$ [55, 41]. 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 Section 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 $\mathbb{F}_{q^n}^*$ 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 essential properties of T_n [41].

Lemma 1. *1. $T_n(\mathbb{F}_q) \cong G_{q,n}$.*
2. $\#T_n(\mathbb{F}_q) = \Phi_n(q)$.
3. 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 $\mathbb{F}_{q^n}/\mathbb{F}_q$.

2.2 The Tate Pairing

In this section we review current algorithms for the computation of the Tate pairing, the body of which is due to the much cited but unpublished work of Miller [33]. It was demonstrated independently by Barreto *et al.* [1] and Galbraith *et al.* [14] that much of the computation of the original algorithm is redundant. In terms of performance, and with a suitable choice of parameters, the recommendations of the former paper provide the better alternative, and we refer to their algorithm as the *reduced* Tate pairing, or the BKLS algorithm. Also, in common with the authors of [44], we refer to the later improvements of Duursma and Lee as the *modified* Tate pairing, but note that the algorithm is essentially a fast method to compute the earlier algorithms of Galbraith *et al.* and Barreto *et al.* Throughout the paper, since we are concerned primarily with pairing-based systems in characteristic three, where relevant we write exponents in ternary.

The reduced Tate pairing. We first introduce some notation. 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 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 [1]. For a thorough treatment of the following, we refer the reader to [1] and also [14], and to [47] for an introduction to divisors. The reduced Tate pairing of order l is the map

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

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:

- For each $P \neq \mathcal{O}_E$ there exists $Q \in E(\mathbb{F}_{q^k})[l]$ such that $e_l(P,Q) \neq 1 \in \mathbb{F}_{q^k}^*/(\mathbb{F}_{q^k}^*)^l$ (*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 $e_l(P,Q)^{(q^k-1)/l} = e_L(P,Q)^{(q^k-1)/L}$.
- It is efficiently computable.

The non-degeneracy condition requires that Q is not a multiple of P, i.e. that Q is in some order l subgroup of $E(\mathbb{F}_{q^k})$ disjoint from $E(\mathbb{F}_q)[l]$. When one computes $f_{P,l}(\mathcal{D})$, the value obtained belongs to the quotient group $\mathbb{F}_{q^k}^*/(\mathbb{F}_{q^k}^*)^l$, and not $\mathbb{F}_{q^k}^*$. In this quotient, for a and b in $\mathbb{F}_{q^k}^*$, $a \sim b$ if and only if there exists $c \in \mathbb{F}_{q^k}^*$ 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 $\mathbb{F}_{q^k}^*/(\mathbb{F}_{q^k}^*)^l$ and the elements of order l in $\mathbb{F}_{q^k}^*$ given by this exponentiation makes it possible to compute $f_{P,l}(Q)$ rather than $f_{P,l}(\mathcal{D})$ [1]. It also removes the need to compute the costly denominators in Miller's algorithm.

The modified Tate pairing. Duursma and Lee introduced their algorithm in the context of pairings on a family of hyperelliptic curves. Restricting to the elliptic curve case, it applies to a family of supersingular curves in characteristic three, including those in Figure 1, 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

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 pairing value embeds, which is always a degree six extension in these cases. 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 on 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.

3 The Quotient Group

In this section we demonstrate that the quotient group to which a Tate pairing output belongs allows one to obtain unique representatives easily, permits fast multiplication, and provides automatic compression by a factor of two.

In this context a quotient group is just the multiplicative group of a finite field modulo a subgroup, and hence is the set of orbits of the first group under multiplication by elements of the subgroup. In particular, the output of the Tate pairing of order q^3+1 is an element of the quotient group

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

For any $a \in \mathbb{F}_{q^6}^*$ we have $a^{q^3+1} \in \mathbb{F}_{q^3}^*$, and so $\mathcal{G}$ simplifies to $\mathbb{F}_{q^6}^*/\mathbb{F}_{q^3}^*$. Hence for $e = f_P(\phi(Q)) \in \mathcal{G}$, multiplication by an element of $\mathbb{F}_{q^3}^*$ does not change the coset represented by e. Let $G_l \subset \mathbb{F}_{q^6}^*$ denote the subgroup of order l. The two properties

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

imply that $e = gh$ for some $g \in G_l$, $h \in \mathbb{F}_{q^3}^*$. 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 $\mathbb{F}_{q^6}$.

Alternatively one can do the following. Let $\mathbb{F}_{q^6} = \mathbb{F}_{q^3}[\sigma]/(\sigma^2+1)$ which is the extension we use for the Duursma-Lee algorithm. Writing $e = e_0 + e_1\sigma$ and $g = g_0 + g_1\sigma$, by the above we have $e = gh = g_0h + g_1h\sigma$. Since the represented coset remains invariant under multiplication by elements of $\mathbb{F}_{q^3}^*$, 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 $\mathbb{F}_{q^3}$. To power by q^3-1 involves a division in $\mathbb{F}_{q^6}$ since

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

so the saving is not significant. 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; the arithmetic is identical to that of $\mathbb{F}_{q^6}$. If one assumes cubings and additions are essentially free, then for whatever practical method one uses to exponentiate, this method will always be roughly one third faster. The defining property of the quotient group $\mathcal{G}$ thus reduces the cost of basic arithmetic.

3.1 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 $\mathbb{F}_{q^3}^*$ 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 $\lambda \in \mathbb{F}_{q^3}^*$ 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* $\lambda \in \mathbb{F}_{q^3}^*$.

$\mathcal{G}_A$ *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 the mixed addition method for point multiplication on elliptic curves. The use of signed digit representations follows since as we show below inverses are cheap, and in Section 4 we derive an exponentiation algorithm using a split exponent method. Since the group operation is essentially multiplication in the field $\mathbb{F}_{q^6}$ we write the operation multiplicatively.

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 $|\mathbb{F}_{q^6}^*/\mathbb{F}_{q^3}^*| = (q^6-1)/(q^3-1) = q^3 + 1$, 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 expolit these observations in the exponentiation algorithms developed in Section 4.

3.2 An equivalent representation of the quotient group

The above method may appear to be an *ad hoc* trick to speed up exponentiation. In this next section we give an alternative interpretation of $\mathcal{G}$ as an algebraic torus. This viewpoint allows a compression of pairing values by a factor of three rather than two, without any further computation.

Given $e = f_P(\phi(Q))$ 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 $\mathbb{F}^*_{q^3}$. This may seem strange 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

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

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 $\mathbb{F}^*_{q^3}$ 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. To see this 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 using the fact that $\sigma^2 + 1 = 0$, we observe 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 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 the quotient group without powering equivalently as operating purely on the denominator of (1) after powering, and so all the arithmetic carries over unchanged. We note that this method also works for any quadratic extension.

Moreover, the fraction (1) is actually a compressed representation of the algebraic torus $T_2(\mathbb{F}_{q^3})$, as we now demonstrate.

Lemma 3. *There is an isomorphism*

$$\tau : T_2(\mathbb{F}_{q^3}) \xrightarrow{\sim} \left\{ \frac{b-\sigma}{b+\sigma}, b \in \mathbb{F}_{q^3} \right\} \cup \{1\},$$

where for $a = a_0 + a_1\sigma \in T_2(\mathbb{F}_{q^3}) \setminus \{1\}$,

$$\tau(a) = \left(\frac{b-\sigma}{b+\sigma} \right), \tag{2}$$

with $b = -(1 + a_0)/a_1$ *if* $a_1 \neq 0$ *and* $b = 0$ *otherwise.*

Proof. By Definition 1 the torus $T_2(\mathbb{F}_{q^3})$ is the set of elements in $\mathbb{F}_{q^6}$ such that $\mathrm{Ker}[N_{\mathbb{F}_{q^6}/\mathbb{F}_{q^3}}] = 1$, which is just the set of elements that satisfy $a^{q^3+1} = 1$. For all $b \in \mathbb{F}_{q^3}$, we see that $N_{\mathbb{F}_{q^6}/\mathbb{F}_{q^3}}((b-\sigma)/(b+\sigma)) = 1$, and counting also the identity, we have all q^3+1 possible solutions. Solving $a_0 + a_1 x = (b-\sigma)/(b+\sigma)$ for b gives two linear equations, which give the stated relations. Equating the two solutions gives $a_0^2 + a_1^2 = 1$, which is just the condition that a be in the stated kernel.

□

Thus the representation (1) of the embedding $\mathcal{G} \hookrightarrow \mathbb{F}_{q^6}$ given by powering by $q^3 - 1$ is just the representation of $T_2(\mathbb{F}_{q^3})$ given above. The reason it is a compressed representation is that one can specify an element uniquely with $b \in \mathbb{F}_{q^3}$, rather than an element of $\mathbb{F}_{q^6}$. Geometrically, b is related to the classical rational parametrisation of the circle S^1 over $\mathbb{F}_{q^3}$. Let t be the intersection of the a_1- axis and the chord joining $(-1, 0)$ to (a_0, a_1) on the unit circle. Then $b = -1/t$ which gives a one-dimensional parametrisation of the torus T_2 (Figure 2). Of course b is just the value e_0/e_1 belonging to the quotient group.

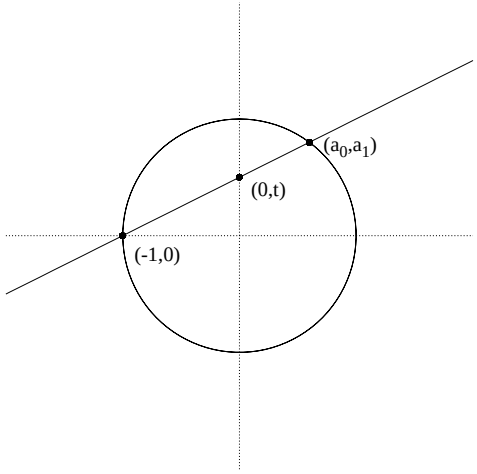

Fig. 2. A rational parametrisation of T_2

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 $T_6(\mathbb{F}_q)$ [22], though we do not rule out such a possibility. We know though that since $|T_6(\mathbb{F}_q)| = (q^2-q+1)|(q^3+1) = |T_2(\mathbb{F}_{q^3})|$

we can use the properties of the latter and apply them to the former, and make use of the improvements derived over the extension field representation.

3.3 Further compression using $T_6(\mathbb{F}_q)$

We have shown that one can easily compress pairings by a factor of two if it is desirable. While one can not easily exploit the additional structure of $T_6(\mathbb{F}_q)$ over $T_2(\mathbb{F}_{q^3})$ to speed up multiplications, one can utilise it for better compression. Rubin and Silverberg showed that since T_6 is birationally isomorphic to $\mathbb{A}^2(\mathbb{F}_q)$ [41], one can map nearly all its elements to the affine plane and use this representation instead for data transmissions.

Complementing this result, in this section we devise a method to compress elements of $T_6(\mathbb{F}_q)$ by a factor of three without mapping to the affine plane at all. Given an element already compressed by a factor of two as in the previous section, our method results in compression by a further factor of $3/2$, for free. A particular benefit of our method is that since we do not have to map to $\mathbb{A}^2(\mathbb{F}_q)$, we do not miss any points by compressing, as is inevitable with the original parametrisation of T_6 given in [41].

Since $T_6(\mathbb{F}_q) = \mathrm{Ker}(N_{\mathbb{F}_{q^6}/\mathbb{F}_{q^3}}) \cap \mathrm{Ker}(N_{\mathbb{F}_{q^6}/\mathbb{F}_{q^2}})$, to obtain a suitable representation we just need to parametrise those elements of the form (2) which have norm equal to one in the second factor.

As in (2) let $b = b_0 + b_1\rho + b_2\rho^2$ where $\rho^3 - \rho \pm 1 = 0$ defines the cubic extension we later use for the pairing computation. Then we obtain an equation in b_0, b_1, and b_2 by the condition

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

This is equivalent to $1+b_1^2-b_0b_2-b_2^2 = 0$, which one can parametrise easily with just b_1 and b_2, since $b_0 = (1+b_1^2-b_2^2)/b_2$. It is therefore sufficient to specify only b_1 and b_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 $\psi : \mathbb{F}_q \times \mathbb{F}_q^* \to T_6(\mathbb{F}_q) \setminus \{1\}$ given by

$$\psi(b_1, b_2) = \frac{((1+b_1^2-b_2^2) + b_1b_2y + b_2^2y^2) + b_2x}{((1+b_1^2-b_2^2) + b_1b_2y + b_2^2y^2) - b_2x},$$

The inverse map $\psi^{-1} : T_6(\mathbb{F}_q) \setminus \{1\} \to \mathbb{F}_q \times \mathbb{F}_q^*$ is given as above, i.e. we just take the second and third coefficients in the fractional expression for $\tau(a)$, with $a \in T_6(\mathbb{F}_q)$. Note that $\mathbb{F}_q \times \mathbb{F}_q^*$ and $T_6(\mathbb{F}_q) \setminus \{1\}$ both have cardinality $q^2 - q$. In terms of $\mathcal{G}$, this means that once e_0/e_1 is computed one can use the second and third coefficients to parametrise the element, which involves no further computation.

Remark. In the context of compression, in [40] it was shown how one can compress BLS short-signatures [5] by using 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 [25]. However, building upon an idea of Semaev [45], Gaudry has shown that such curves are weaker than those defined over a field with prime extension degree [17]. 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.

3.4 Application to characteristic two supersingular curves

For suitable supersingular elliptic curves in characteristic two the embedding degree is four. Here one can use $T_2(\mathbb{F}_{q^2})$, although there is no correspondingly simple version of the Duursma-Lee algorithm for the Tate pairing. This means part of the final powering needs to be computed as follows.

Let $e \in \mathbb{F}^*_{q^4}/(\mathbb{F}^*_{q^4})^l$ be the output of the Tate pairing where $l = q + 1 \pm \sqrt{2q}$. To obtain a unique value one needs to raise e to the power $(q + 1 \mp \sqrt{2q})(q^2 - 1)$. Performing an exponentation by just the first term, which costs just one application of the q-Frobenius, a few squarings and two multiplications, the resulting element then belongs to the quotient group $\mathbb{F}^*_{q^4}/(\mathbb{F}^*_{q^4})^{q^2+1} = \mathbb{F}^*_{q^4}/\mathbb{F}^*_{q^2}$. This allows us to ignore the powering by $(q^2 - 1)$ as before and one can then apply to this group all the methods we have developed for $\mathcal{G}$, including fast exponentiation and compression by a factor of two.

3.5 Application to MNT Curves

Miyaji, Nakabayashi and Takano have described an efficient method for the generation of non-supersingular elliptic curves with small embedding degree [34]. As security requirements adapt to growing computational power, the optimal embedding degree will rise from the current recommendation of six to perhaps twelve or even higher. Since the embedding degree available for supersingular curves is restricted to six, in the long term such curves will likely become prominant in the deployment of identity-based technologies.

Letting the embedding degree be $2k$, which is always the case in practice [1], we can again use the quotient group $\mathcal{G}$ to simplify the final powering, and to speed up later arithmetic operations on a pairing value.

One needs to compute the Tate pairing of order l, where $l|\Phi_{2k}(q)$. To obtain a unique representative one needs to raise the output e by the power $(q^{2k}-1)/l$. In exactly the same way as for even characteristic we raise by the power $(q^k+1)/l$ to obtain an element in the the quotient group $\mathbb{F}^*_{q^{2k}}/\mathbb{F}^*_{q^k}$ and can apply the methods of $\mathcal{G}$ once again.

In large characteristic there are particularly fast methods for exponentiation using special field representations when the extension degree is a multiple of six [51], or indeed a fast version of XTR [52]. If the embedding degree is even, then one has the choice of either using $\mathcal{G}$ or LUC [49], the latter being faster on account of squaring or cubing no longer being free. However given that the computation of the Tate pairing on general MNT curves with current algorithms

is orders of magnitude slower than in characteristic three, whether to use large characteristic curves or not is still an issue requiring further research.

4 Exponentiation

In this section we describe fast algorithms for exponentation in $\mathcal{G}$, $\mathbb{F}_{q^6}$ and point multiplication in $E(\mathbb{F}_q)$. 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 [7], 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 [32][Chapter 14, Algorithm 14.85]. To do so one needs to precompute and store e^i for $0 < i < 27$. Note that in this method and the previous one all precomputed values for which $i = 0 \bmod 3$ can be computed very cheaply.

Method 4: Frobenius expansion

For $e \in \mathcal{G}$ the q-Frobenius map is easily computed. Moreover, the q-th power of a reduced element is reduced itself. Since the Frobenius map satisfies $q^2 - q + 1 = 0$ 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 $k \equiv k_1 + k_2 q \mod l$ [51]. 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 [53]. 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 [15].

We note that for supersingular curves over characteristic three there is also an efficient scalar multiplication algorithm due to Koblitz [26] 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

$$(e_0/e_1 + \sigma)^{i+qj} \quad i, j \in \{0, \pm 1, \pm 2\} \tag{3}$$

in affine. Let (i, j) represent the corresponding term in (3). Then we can use the fact that for any $e \in \mathcal{G}$, we have $e^{q^2-q+1} = \mathcal{O}_{\mathcal{G}}$ and that the Frobenius map of an affine element is cheap and still affine, 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 $\mathbb{F}_{q^6}$ we use the same method, having first powered e by $q^3 - 1$, but clearly without needing to obtain affine representatives.

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 [14, 23] and represent

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

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

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

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

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

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

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

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

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

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

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

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

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 [29] 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 $t_{00} = a_0^2$, $t_{11} = a_1^2$, $t_{22} = a_2^2$, $t_{01} = a_0 a_1$, $t_{12} = a_1 a_2$, $t_{20} = a_2 a_0$, 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 $t_{00} = c_0 d_0$, $t_{11} = c_1 d_1$, 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 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 $\phi(x_2, y_2) = (\rho - x_2, \sigma y_2)$ (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

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 $18M$, in [44] it was claimed that besides the necessary cubings and cube roots, each loop iteration costs $20M$. However, in each iteration g is sparse, and this can be exploited to reduce the cost of multiplying g and f, which is not sparse in general, to $13M$ (we thank Keith Harrison for pointing this out to us). This total of $15M$ improves on the trace-based method suggested by Scott and Barreto and hence the proposed method does not seem to provide any improvement. 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. If one performs the same loop unrolling for the trace method of [44], the first stage equals $6M+4M$ also, however the equivalent of multiplying f and g then costs $27M$, and so there seems no way to make this method as fast as the ordinary field multiplication proposed here.

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 [16, 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 [39, 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. We do not quote results from our work in this area, preferring to leave a more thorough investigation for further work.

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.

6.1 Comparison with trace-based method

We have shown that the Duursma-Lee algorithm can be implemented very efficiently. We consider the cost of a typical protocol run consisting of a pairing evaluation and an exponentiation in $\mathcal{G}$.

The cost of a mixed multiplication in $\mathcal{G}$ is $12M$. Since $l \approx 3^m$, for an exponentiation 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 even naive field arithmetic is better than the proposed trace method, which in fact can be reduced further to about $10.3mM$ using a Euclidean algorithm, but is still slow [52].

With regard to the trace-based proposal for computing the Tate pairing, ignoring the final powering and assuming the same pre-computation strategy for cube roots for the Duursma-Lee algorithm, a simple cost estimate of an entire protocol run as described amounts to $29mM + (20/3)mC$. For our algorithms, the corresponding cost is just $18.5mM + 5mC$ plus a small amount of pre-computation. In summary therefore we combine the fastest known method for the Tate pairing evaluation, and also later exponentiations.

Remark. In [44], an open problem was suggested 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 $p > 3$. For pairing-based applications, the desirable extension degrees in the near future are likely to

remain small, and no larger than twenty. Since the maximum compression factor possible for a given extension degree n is $n/\phi(n)$, assuming one has compression mechanisms that achieve these values, for $n < 20$ the maximum possible is only three, which is already available.

With respect to the problem proposed, whether the computation of the pairing is in compressed form or not is not particularly relevant unless one is computing in a memory-constrained environment. The computation of the compressed pairing in [44] is in fact not compressed at all, since it requires a ladder of three elements of $\mathbb{F}_{q^2}$ to be held throughout, and thus is equivalent to storing an element of $\mathbb{F}_{q^6}$. The memory footprint of the computation is even higher. The issue of compression should therefore be restricted to communication and not computation, and for both ours and the trace-based system, the compression factor is clearly the same, and is optimal for all practical purposes.

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 3 shows the result of timing this implementation using a variety of different base field sizes. In the pairing section, Algorithms 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 4 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 $\mathbb{F}_{q^6}$. Contrary to a claim of Koblitz [26] 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 of a cryptographic size, this value is in fact closer to 1.3, when using $\mathcal{G}$. This is a strong result since one may have inferred from the fact that the extension field is six times larger than the field of definition for E, that exponentiations in $\mathbb{F}_{q^6}$ are bound to be significantly inferior. Combining all our methods, we see this is not the case. 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.

Fig. 3. 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 $\mathbb{F}_{q^6}$						
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	$1.18ms$	$1.95ms$	$8.11ms$	$12.75ms$	$19.04ms$	$55.93ms$

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 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 which improve upon other suggestions for its implementation. 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 [50]. 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.

Fig. 4. 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	$75.70\mu s$	$106.10\mu s$	$227.10\mu s$	$296.80\mu s$	$347.30\mu s$	$745.10\mu s$

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 [8, 9]. Preliminary research into this problem using Adleman's Function Field Sieve has been conducted [20, 21] but the problem should still be considered open.

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

Acknowledgements

The authors would like to thank Paulo Barreto, Steven Galbraith, Keith Harrison, Mike Scott, Nigel Smart and Fré Vercauteren for many helpful comments on an early draft of this article.

References

1. 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.
2. 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.

3. D. Boneh, X. Boyen and H. Shacham. Short Group Signatures. To appear in *Advances in Cryptology (CRYPTO 2004)*.
4. D. Boneh and M. Franklin. Identity-Based Encryption from the Weil Pairing. In *SIAM Jounal on Computing*, Volume 32, no. 3, 586-615, 2003.
5. 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.
6. W. Bosma, J. Hutton and E. Verheul. Looking beyond XTR. In *Advances in Cryptology (ASIACRYPT 2002)*, Springer LNCS 2501, 46–63, 2002.
7. L. Chen and J. Malone-Lee. Improved Identity-Based Signcryption. Cryptology ePrint Archive, Report 2004/114. Available from `http://eprint.iacr.org/2004/114`.
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. D. Coppersmith. Evaluating logarithms in GF(2^n). In *16th ACM Symp. Theory of Computing*, 201–107, 1984.
10. D. Coppersmith. Fast evaluation of logarithms in fields of characteristic two. *IEEE Trans. Info. Theory*, **30**, 587–594, July 1984.
11. 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.
12. T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *IEEE Trans. Info. Theory* **31**, 469–472, 1985.
13. 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.
14. S. Galbraith. Supersingular Curves in Cryptography. In *Advances in Cryptology (ASIACRYPT 2001)*, Springer LNCS 2248, 495–513, 2001.
15. S. Galbraith, K. Harrison and D. Soldera. Implementing the Tate pairing. In *Proc. of ANTS V*, Springer LNCS 2369, 324–337, 2002.
16. 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.
17. S. Gao. Normal Bases over Finite Fields. PhD Thesis, Waterloo University, 1993.
18. 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`.
19. C. Gentry. Certificate-Based Encryption and the Certificate Revocation Problem. In *Advances in Cryptology (EUROCRYPT 2003)*, Springer LNCS 2656, 272–293, 2003.
20. P Golle and A. Juels. Dining Cryptographers Revisited. In *Advances in Cryptology (EUROCRYPT 2004)*, Springer LNCS 3027, 456–473, 2004.
21. R. Granger. Estimates for discrete logarithm computations in finite fields of small characteristic. In *Cryptography and Coding*, Springer LNCS 2898, 190–206, 2003.
22. R. Granger, A. Holt, D. Page, N. Smart and F. Vercauteren. Function Field Sieve in Characteristic Three. To appear in *Proc. of ANTS VI*.
23. R. Granger, D. Page and M. Stam. A Comparison of CEILIDH and XTR. To appear in *Proc. of ANTS VI*.
24. 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.

25. F. Hess. Efficient Identity based Signature Schemes based on Pairings. In *Selected Areas in Cryptography (SAC 2002)*, Springer LNCS 2595, 310–324, 2003.
26. A. Joux. A One Round Protocol for Tripartite Diffie-Hellman. In *Proc. of ANTS IV*, Springer LNCS 1838, 385–394, 2000.
27. N. Koblitz. An elliptic curve implementation of the finite field digital signature algorithm. *Advances in Cryptology (CRYPTO 98)*, Springer LNCS 1462, 327–337, 1998.
28. A. Lenstra. Using Cyclotomic Polynomials to Construct Efficient Discrete Logarithm Cryptosystems over Finite Fields. In *Proc. of ACISP97*, Springer LNCS 1270, 127–138, 1997.
29. A. Lenstra and E. Verheul. The XTR Public Key System. In *Advances in Cryptology (CRYPTO 2000)*, Springer LNCS 1880, 1–19, 2000.
30. 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.
31. A. Menezes, T. Okamoto, and S. A. Vanstone. Reducing Elliptic Curve Logarithms to Logarithms in a Finite Field. In *IEEE Trans. Info. Theory*, **39**, 1639–1646, 1993.
32. A.J. Menezes, P.C. van Oorschot and S.A. Vanstone. Handbook of Applied Cryptography. CRC Press, 1997.
33. V. Miller. Short programs for functions on curves. Unpuplished manuscript, 1986. Available from `http://crypto.stanford.edu/miller/miller.pdf`.
34. A. Miyaji, M. Nakabayashi and S. Takano. New explicit conditions on elliptic curve traces for FR-reduction. In *IEICE Trans. Fundamentals* E-84 A(5), 1234-1243, 2001.
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. K. Paterson. ID-based signatures from pairings on elliptic curves. Cryptology ePrint Archive, Report 2002/004. Available from `http://eprint.iacr.org/2002/004`.
38. 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.
39. 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.
40. K. Rubin and A. Silverberg. Supersingular abelian varieties in cryptology. In *Advances in Cryptology (CRYPTO 2002)*, Springer LNCS 2442, 336–353, 2002.
41. K. Rubin and A. Silverberg. Torus-Based Cryptography. In *Advances in Cryptology (CRYPTO 2003)*, Springer LNCS 2729, 349–365, 2003.
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. Cryptology ePrint Archive, Report 2004/032. Available from `http://eprint.iacr.org/2004/032`.

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. A. Shamir. Identity-Based Cryptosystems and Signature Schemes. In *Advances in Cryptology (CRYPTO '84)*, Springer LNCS 196, 47–53, 1985.
47. J. Silverman. The arithmetic of elliptic curves. Springer GTM 106, 1986.
48. N. Smart. Access control using pairing based cryptography. In *Proceedings CT-RSA 2003*, Springer LNCS 2612, 111–121, 2003.
49. 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.
50. J.A. Solinas. Low-Weight Binary Representations for Pairs of Integers. University of Waterloo, Technical Report CORR 2001-41.
51. 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.
52. M. Stam and A. Lenstra. Speeding Up XTR. In *Advances in Cryptology (ASIACRYPT 2001)*, Springer LNCS 2248, 125–143, 2001.
53. E.G. Straus. Problems and Solutions: (5125) Addition Chains of Vectors. In *American Mathematical Monthly*, **71**, 806–808, 1964.
54. E. Verheul. Self-blindable credential certificates from the Weil pairing. In *Advances in Cryptology (ASIACRYPT 2001)*, Springer LNCS 2248, 533–551, 2001.
55. V.E. Voskresenskiĭ. Algebraic Groups and Their Birational Invariants. In *Translations of Mathematical Monographs*, **179**, American Mathematical Society, 1998.