A comparison of MNT curves and supersingular curves

D. Page, N.P. Smart and F. Vercauteren

Dept. Computer Science,
University of Bristol,
Merchant Venturers Building,
Woodland Road,
Bristol,
BS8 1UB, United Kingdom.
{page,nigel,frederik}@cs.bris.ac.uk

Abstract. We compare both the security and performance issues related to the choice of MNT curves against supersingular curves in characteristic three, for pairing based systems. We pay particular attention to equating the relevant security levels and comparing not only computational performance and bandwidth performance. The paper focuses on the BLS signature scheme and the Boneh–Franklin encryption scheme, but a similar analysis can be applied to many other pairing based schemes.

1 Introduction

The initial pairing based cryptographic protocols, see [15], [22] and [23], were couched in the language of a generic pairing function. However, in latter work such as the (BLS) short signature scheme of Boneh, Lynn and Shacham [8] or the identity based encryption scheme of Boneh and Franklin [7] the protocols are given in terms of a symmetric pairing, as is found on supersingular curves.

For all protocols there are essentially two possible choices for selecting the underlying groups on which to work:

- To use supersingular curves, of which the most efficient ones appear to be in characteristic three.
- To use, so-called, MNT curves in large prime characteristic [19]. These are ordinary, as opposed to supersingular curves.

There are a number of advantages in using supersingular curves. For example they possess distortion maps which makes various protocols possible [6] or enable proofs to work [5], in addition supersingular curves have very efficient algorithms in characteristic three for computing the Tate pairing [10] and the underlying field arithmetic can be implemented relatively efficiently. On the negative side however, one could question their long term security due to the fact that they are supersingular, because there are very few usable curves with the correct properties or because of more efficient algorithms for the discrete logarithm problem in fields of small characteristic, see [9] and [12] for a discussion relevant to characteristic three.

Whilst ordinary elliptic curves do not exhibit the disadvantages mentioned above they do present other issues. For example, distortion maps no longer exist and one needs to very carefully select how one implements the various protocols and not simply the Tate pairing. Protocols such as [5] and [6] which require distortion maps can be implemented in the setting of MNT curves via the use of the trace map, however this requires a non-optimal choice of the group G_2 below leading to a much more inefficient pairing algorithm. We do not address this issue further in this paper.

1.1 Prior Work and Notation

Let G_1, G_2 and G_T denote finite abelian groups in which the discrete logarithm problem is hard. By a pairing we shall mean a non-degenerate bilinear map

$$\hat{t} : G_1 \times G_2 \longrightarrow G_T.$$

This is usually defined via the modified Tate pairing $\hat{t}(\cdot, \cdot)$ on an elliptic curve. We let $G_1 = E(\mathbb{F}_q)$ denote an elliptic curve over the finite field $\mathbb{F}_q$ with order divisible by N such that N also divides $q^\alpha - 1$, where α is the order of q in $\mathbb{Z}_N^*$ and is called the MOV embedding degree. If $P \in E(\mathbb{F}_q)$ we let f_P denote the function with divisor

$$(f_P) = N(P) - N(\mathcal{O}).$$

If $Q \in \mathbb{F}_{q^\alpha}$ we define the unmodified Tate pairing via

$$t(P, Q) = f_P(Q).$$

This can be computed via variants of Miller's algorithm [20] and results in an element of $\mathbb{F}_{q^\alpha}^* / (\mathbb{F}_{q^\alpha}^*)^N$.

The modified Tate pairing takes values in the subgroup G_T of $\mathbb{F}_{q^\alpha}^*$ of order N and is defined in one of two ways, depending on whether E is a supersingular or ordinary (MNT) curve. In the case of supersingular curves we define an integer n via $q = 3^n$, for the case of MNT curves we define m via $q \approx 2^m$. One of our tasks will be to compare the values of n and m, in terms both of security and of protocol efficiency.

Supersingular Case: In this case one defines $G_1 = G_2 = E(\mathbb{F}_q)$ and one lets

$$\hat{t}(P, Q) = t(P, \phi(Q))^{(q^\alpha - 1)/N},$$

where ϕ is a so-called distortion map

$$\phi : E(\mathbb{F}_q) \longrightarrow E(\mathbb{F}_{q^\alpha}).$$

Such a definition of the modified Tate pairing is guaranteed to be non-degenerate, for all $P \in E(\mathbb{F}_q)$ with $P \neq \mathcal{O}$ we have $\hat{t}(P,P) \neq 1$ and in addition for all $P, Q \in E(\mathbb{F}_q)$ we have

$$\hat{t}(P,Q) = \hat{t}(Q,P).$$

This last property means that when defining protocols we do not need to worry about whether a point lies in G_1 or G_2, since both groups are the same and can be treated equally.

Various authors have considered implementation issues for pairing based systems, see [2] and [11]. For supersingular curves the best parameter choices are in characteristic three. In this case one can select $\alpha = 6$, resulting in systems which are secure against discrete logarithm attacks in G_1 and G_2 and also (hopefully) against discrete logarithm attacks in G_T. The most efficient algorithm for implementing the modified Tate pairing in characteristic three is by Duursma and Lee [10]. In addition the point multiplication algorithms in characteristic three can be implemented using affine coordinates, due to the efficient tripling formulae.

Ordinary Case: In this case one selects G_2 to be a subgroup of $E(\mathbb{F}_{q^\alpha})$ and

$$\hat{t}(P,Q) = t(P,Q)^{(q^\alpha - 1)/N}.$$

Note that it is no longer possible to exchange the role of P and Q like in the supersingular case. Furthermore, care needs to be taken as to whether a point lies in G_1 or G_2 when one defines protocols. For example, one of the groups is used to hash strings into, whilst another is used for key definitions.

In [2] various techniques are given to improve performance. The most important being the use of denominator elimination in the pairing algorithm when N does not divide $q - 1$, α is even and $x(Q) \in \mathbb{F}_{q^{\alpha/2}}$. The paper compares such curves with supersingular curves in characteristic three, but is unclear as to which groups one is hashing into and which are being used for curve operations. The paper [14] gives explicit formulae for the pairing computation using projective coordinates. It only considers the issue of actually computing the pairing and only gives estimated operation counts. In addition no comparison is made to supersingular curves. The paper [3] uses the twist of the curve $E(\mathbb{F}_{q^{\alpha/2}})$ to generate the group G_2, and again only looks at the pairing computation.

1.2 Our Contribution

In this paper we present a comparison of large prime MNT curves with supersingular curves in characteristic three. We pay particular attention to how the Boneh–Franklin encryption and BLS signature scheme are defined in this context. We also discuss our parameter and implementation choices. The main goal is to be able to compare the implementation efficiency of the two cases and make sensible recommendations as to security versus efficiency tradeoffs. For example, if supersingular curves are much more efficient than MNT curves yet one is worried about possibly weaker security in using supersingular curves, one could simply take larger parameters so as to guard against any possible threat. We do not explain in any detail our choices in characteristic three since we have adopted standard practice which is well documented elsewhere [11, 13, 10].

2 MNT Parameter Choices

In this section we present our choices for implementing MNT curves and how these choices impact on cryptographic protocols. This is an issue which to some extent is not covered in prior work.

Like [2] and [3] we select α to be even and choose a polynomial basis for $\mathbb{F}_{q^\alpha}$ over $\mathbb{F}_q$ by a irreducible polynomial with no odd terms, i.e.

$$F_{q^\alpha} = \mathbb{F}_q[\chi]/(f(\chi))$$

with

$$f(\chi) = \chi^\alpha + c_{\alpha-2}\chi^{\alpha-2} + \cdots + c_2\chi^2 + c_0.$$

A curve is selected of the form

$$E : y^2 = x^3 - 3x + B$$

where $B \in \mathbb{F}_q$, with group order divisible by a large prime N and such that N divides $q^\alpha - 1$, with N not dividing $q^\beta - 1$ for $\beta < \alpha$. We are particularly interested in balancing security against bandwidth, hence one wishes to select a q which is as small as possible, to minimize bandwidth, whilst choosing N and α large enough to increase security. We feel this is best accomplished by finding curves with $q \approx N$ and with α and N large enough to give the required security level. Due to the current best construction techniques one therefore selects $\alpha = 6$ or $\alpha = 12$ [4]. Other authors choose $\alpha = 2$, this enables them to select curves with values of N with low Hamming weight to improve efficiency. However, this comes at the expense of greater bandwidth and slower arithmetic in $E(\mathbb{F}_q)$, due to the larger value of q required. Choosing $\alpha = 6$ also allows a more direct comparison with the case of supersingular curves in characteristic three. Hence, in the following we focus soley on $\alpha = 6$.

In [3] the group G_2 is defined as the group $\overline{E}(\mathbb{F}_{q^{\alpha/2}})$, where $\overline{E}$ is the quadratic twist of E over $\mathbb{F}_{q^{\alpha/2}}$. The modified pairing is then computed via mapping the element in G_2 over to $E(\mathbb{F}_{q^\alpha})$ before applying the standard pairing. Equivalently, one can simply define G_2 as the subgroup of $E(\mathbb{F}_{q^\alpha})$ defined by

$$G_2 = \left\{(x, y) \in E(\mathbb{F}_{q^\alpha}) : x \in \mathbb{F}_{q^{\alpha/2}}, y \notin \mathbb{F}_{q^{\alpha/2}}\right\}.$$

Arithmetic in G_2 can then be defined simply in terms of arithmetic in $\mathbb{F}_{q^{\alpha/2}}$, since y is of the form $\chi \cdot \gamma$, with $\gamma \in \mathbb{F}_{q^{\alpha/2}}$. Indeed, by definition we have that y^2 is a non-square in $\mathbb{F}_{q^{\alpha/2}}$ and due to the special choice of $f(\chi)$, χ^2 will be a non-square in $\mathbb{F}_{q^{\alpha/2}}$ too. Dividing y^2 by χ^2 thus gives a square in $\mathbb{F}_{q^{\alpha/2}}$, which shows that y is indeed of the form $\chi \cdot \gamma$ with $\gamma \in \mathbb{F}_{q^{\alpha/2}}$. In addition one can use the efficient algorithm for projective doubling in G_2 which uses the nice form of the curve equation for E.

The basic ingredient of both the BLS signature scheme and the Boneh–Franklin encryption scheme is that one of our groups G_i is used to hold standard elliptic curve public/private key pairs

$$(Q = [a]P, a)$$

with $P, Q \in G_i$. The other group $G_{i'}$ is used to map arbitrary messages or identities into. If one was only interested in BLS signatures then bandwidth is determined by the size of the elements in $G_{i'}$, where in the Boneh–Franklin scheme message size is dominated more by the size of elements in G_i. However, one should notice that the user based secret keys in the Boneh–Franklin scheme are nothing but the BLS signature of the trust authority on the identity string. One can then argue that it is likely to be more important to minimize bandwidth for the Boneh–Franklin encryption scheme, since the BLS signature scheme is more likely to be used within the context of the issuing of secret keys for the Boneh–Franklin scheme.

In addition, point multiplication is required in $G_{i'}$ for BLS signature generation and in G_i for Boneh–Franklin encryption and decryption. Hence, it also makes sense to choose G_i and $G_{i'}$ so that point multiplication is cheaper in G_i. This is because it is likely that in an identity based system more messages are encrypted/decrypted than identity based keys issued.

Hence, in our implementation we make use of a hash function

$$H : \{0,1\}^* \longrightarrow G_2.$$

Given our definition for G_2 we need only map the string to an element of $\mathbb{F}_{q^{\alpha/2}}$ plus an associated choice for the y-coordinate. In transmitting elements of G_2 we only need to transmit the x-coordinate, requiring $(\alpha \log_2 q)/2$ bits, plus a bit for the y-coordinate.

One problem with this choice is that it makes BLS signature generation more expensive as one needs to perform a point multiplication in G_2, which is more expensive than a point multiplication in G_1. However, the advantage is that Boneh–Franklin encryption can be performed via point multiplication in G_1 and message sizes for Boneh–Franklin encryption are smaller. We stress that if one was soley interested in BLS signatures then one would swap the roles of G_1 and G_2.

One can even remove the need entirely for passing the y-coordinate in both G_1 and G_2: by not transmitting any y-coordinates of any points the receiver knows the point Q only up to $\pm Q$. A careful examination of most of the major pairing based protocols reveals that one can allow for this ambiguity as follows: when one computes $\hat{t}(P, Q)$, if one only knows P or Q or both up to a sign then the resulting pairing value is equal to

$$z = \hat{t}(\pm P, \pm Q) = \hat{t}(P, Q)^{\pm 1}.$$

Hence, computing

$$z' = z + \frac{1}{z}$$

removes all ambiguity and one then uses z' in place of z in the definition of the Boneh–Franklin scheme. Note, a simpler solution applies to the BLS scheme which requires no inversion of the value z. Removing the need for passing y, or more correctly the compression of y, could enable the avoidance of various patents on point compression.

To see this in more detail consider the BLS signature scheme defined as follows:

Key Generation: Let P denote a generator of G_1 and let $a \in \mathbb{Z}_N^*$ denote the secret key. The public key is $v = x(V)$, where $V = [a]P$ and $x(V)$ denotes the x-coordinate of the point V.

Signature Generation: Compute $U = H(M) \in G_2$, the signature is $s = x(S)$ where $S = [a]U$.

Signature Verification: Compute $U = H(M) \in G_2$, and from v and s recover $\pm V$ and $\pm S$. Compute

$$r_1 = \hat{t}(P, \pm S) \text{ and } r_2 = \hat{t}(\pm V, U).$$

Accept the signature if and only if $r_1 = r_2$ or $r_1 \cdot r_2 = 1$. This verification works since

$$r_1 = \hat{t}(P, [a]U)^{\pm 1} = \hat{t}([a]P, U)^{\pm 1} = r_2.$$

The resulting protocol is still provably secure.

One can also apply the BLS scheme with the roles of G_1 and G_2 reversed. In this case a signature lies in G_1, whilst a public key lies in G_2. This case is to be preferred if one is not using BLS signatures as credentials in a Boneh–Franklin scheme, as in [1] and [18], but one is using them on their own to achieve signatures with a small size. Such a scheme we shall denote by BLS$^\perp$. Note, that the difference between BLS and BLS$^\perp$ only occurs for MNT curves, since in this situation $G_1 \neq G_2$.

3 MNT Curves Used

Curves with $q \equiv 3 \pmod 4$ are to be preferred since for these we have efficient square root algorithms. If one needs to select a q such that $q \equiv 1 \pmod 4$, then select one such that $q^{\alpha/2} \equiv 5 \pmod 8$ for a similar reason.

The curves we selected are given in the Appendix; most of them have cofactor 1, which by the MNT construction [19] implies that $q = 4l^2 + 1$ and $t = 1 \pm 2l$ for some integer $l \in \mathbb{Z}$. The integer l satisfies $x = 6l \mp 1$ where x is a solution to the generalized Pell equation

$$x^2 - 3Dy^2 = -8\,,$$

with D the discriminant of the elliptic curve, i.e. the square-free part of $4p - t^2$.

To generate suitable curves we solved this equation for every discriminant $D \leq 3 \cdot 10^9$ and $50 \leq |x| \leq 300$. Note that two trivial observations speed up this process considerably: an easy analysis as in [19] shows that $D \equiv 3 \pmod 8$ and reducing the equation modulo $3D$ shows that -8 should be a square modulo $3D$. To solve the generalized Pell equation we modified the algorithm described in [21] to also take into account that only relatively small x, i.e. less than 300 bits, are useful. The curves with cofactor 2 were found using the generalised construction described in [26].

Given the discriminant D and the prime q, we used Mike Scott's implementation [25] of the Complex Multiplication algorithm to generate the equation of the curve. For large D and q however, this program failed and the curves were kindly generated by Andreas Enge.

4 Security Comparison

In comparing MNT curves with supersingular curves in characteristic three, from a security perspective one only needs to consider the relative difficulty of the discrete logarithm problem in the respective groups G_T. This is because the best known algorithms to solve the discrete logarithm in G_1 and G_2 are the same in both situations. As before we let q denote the size of the base field for the elliptic curve defining G_1, hence one is interested in solving for discrete logarithms in $\mathbb{F}_{q^\alpha}^*$. Due to special purpose discrete logarithm algorithms in characteristic two, such as Coppersmith's algorithm [9] or the function field sieve [16], some researchers have questioned the use of low characteristic fields for pairing based cryptosystems.

It is certainly easier to implement the respective discrete logarithm algorithm in smaller characteristic. For example the record for discrete logarithms in characteristic two is in the field $\mathbb{F}_{2^{607}}$ [27], whereas the record for large prime characteristic is for $q \approx 2^{398}$, [17]. It must however be stated that no data points are available for computing discrete logarithms in $\mathbb{F}_{p^6}$ where p is a large prime.

The complexity of the Number Field Sieve algorithm to solve for discrete logarithms in large characteristic fields is given by [24]

$$L_{q^\alpha}\left(1/3, (64/9)^{1/3}\right) = \exp\left(\left((64/9)^{1/3} + o(1)\right)(\log q^\alpha)^{1/3}(\log\log q^\alpha)^{2/3}\right).$$

The complexity of the function field sieve in low characteristic fields is given by [16]

$$L_{q^\alpha}\left(1/3, (32/9)^{1/3}\right) = \exp\left(\left((32/9)^{1/3} + o(1)\right)(\log q^\alpha)^{1/3}(\log\log q^\alpha)^{2/3}\right).$$

If we let q_3 denote the size of q for supersingular curves in characteristic three and q_p denote the size of q for MNT curves in large prime characteristic p, then if we wish to have a greater security margin for supersingular curves then we require

$$2(\log q_p^6)(\log\log q_p^6)^2 \leq (\log q_3^6)(\log\log q_3^6)^2.$$

Setting $q_3 \approx q_p^t$ for some constant t, which we wish to determine, leads to the inequality

$$2(\log q_p^6)(\log\log q_p^6)^2 \le t(\log q_p^6)(\log t + \log\log q_p^6)^2$$

This simplifies to

$$2(\log\log q_p^6)^2 \le t(\log t + \log\log q_p^6)^2.$$

Now for fields under consideration, i.e. $2^{1000} \le q_p^6 \le 2^{2000}$ we have $\log\log(q_p^6) \approx 7$ and so we should take $t \approx 1.7$. Hence, setting

$$q_3 \approx q_p^{1.7}$$

would be a conservative estimate for the security of supersingular curves in characteristic three compared to MNT curves, both with MOV parameter $\alpha = 6$. Note, this is very conservative towards the security of supersingular curves. For example the current records for discrete logarithms in characteristic two and large prime characteristic, see [27] and [17], we see that the ratio is $t \approx 607/398 \approx 1.53$. However, since the NFS algorithm in extension fields of degree six of large prime characteristic base fields, see [24], is more complicated than the NFS algorithm in finite fields of extension degree one, we feel that this conservative estimate is justified.

So if $q_3 = 3^n$ and $q_p \approx 2^m$ then we have

$$n \approx 1.07m$$

In the appendix we give five MNT curves of varying security levels. In Table 1 we present the various security parameters for the curves chosen, this table enables one to compare security of the MNT curves and the supersingular curves. We let $SS(k, \pm)$ denote the supersingular curve over $\mathbb{F}_{3^k}$ given by

$$Y^2 = X^3 - X \pm 1.$$

The column $s_{\mathtt{ECC}}$ refers to the ECC security parameter, namely the bit size of the largest prime subgroup of G_1. The column $s_{\mathtt{RSA}}$ refers to the equivalent RSA-style security parameter. For MNT curves this is equal to $6 \cdot s_{\mathtt{ECC}}$, since by convention one assumes that the security level for discrete logarithms in a finite field of large prime characteristic is equivalent to that of RSA, and the size of the finite field G_T is $6 \cdot s_{\mathtt{ECC}}$ bits.

For the supersingular curves we estimate the security via the above analysis as

$$s_{\mathtt{RSA}} \approx (6 \cdot n)/(1.07) \approx 5.6n.$$

Note, that this is a very cautious approximation from the point of view of using supersingular curves in characteristic three. Not only due to the above analysis, but also since arithmetic in characteristic three is computationally more involved than characteristic two or large prime arithmetic for a computer. Hence, the implied constants in the FFS and NFS big-O notation is likely to be relatively larger for characteristic three than a simple analysis as above would imply.

We therefore see that we should compare our MNT Curve B with the supersingular curve $SS(193, -)$, MNT Curve C with the supersingular curve $SS(239, -)$ etc.

Table 1. Curve Security Comparison

Curve	$s_{\mathtt{ECC}}$	$s_{\mathtt{RSA}}$
$SS(97,+)$	151	845
Curve A	160	960
$SS(163,-)$	258	912
Curve B	191	1146
$SS(193,-)$	305	1080
Curve C	221	1326
$SS(239,-)$	379	1338
Curve D	256	1536
Curve E	307	1842
$SS(353,-)$	559	1976

5 Efficiency Comparison

We compared the timings obtained for the above five MNT curves against comparable supersingular curves in characteristic three. Not only did we time the underlying curve arithmetic and the Tate pairing computation time, but we also timed various suboperations of BLS signatures and Boneh–Franklin encryption.

All timings are given in milli-seconds and are generated on a Windows XP machine with a Pentium 4 running at 2.40 GHz and 256 MB RAM. In Table 2 we give the timings for point multiplication in the groups G_1 and G_2, where the multiplier is an integer of l bits, we also give the time needed to compute a pairing between an element of G_1 and an element of G_2. In Table 3 we present timings for the various cryptographic operations in the schemes we are concentrating on. In Table 4 we present the bandwidth considerations for the various schemes, in terms of public key size, private key size and the message itself. In the case of Boneh–Franklin encryption we assume that the scheme is used to encrypt a message of l bits in length, using a block cipher with key length $m \approx \log_2 q$ bits. This is a standard modification of the Boneh–Franklin scheme.

With this comparison we see that supersingular curves appear less efficient than using MNT curves, for a similar security parameter. However, this conclusion depends precisely on which schemes one wishes to implement and whether one is interested in bandwidth or computational efficiency, or both. Signature generation with BLS is more efficient with supersingular curves, however verification is more expensive. For Boneh–Franklin encryption the times for the MNT curves are significantly more efficient.

6 Acknowledgements

The authors would like to thank Dan Boneh, Sanjit Chatterjee and Steven Galbraith for comments on an earlier draft of this manuscript.

Table 2. Timings for Curve Operations (ms)

	G_1		G_2		Tate
Curve	Affine	Proj	Affine	Proj	Pairing
$SS(97,+)$	5	4	5	4	17
Curve A	9	2	107	65	33
$SS(163,-)$	19	15	19	15	57
Curve B	15	5	151	94	56
$SS(193,-)$	23	22	23	22	86
Curve C	21	7	207	142	80
$SS(239,-)$	42	38	42	38	124
Curve D	30	9	327	201	108
Curve E	50	16	538	353	194
$SS(353,-)$	108	94	108	94	355

Table 3. Timings for Cryptographic Operations (ms)

	BLS Signatures			BLS$^{\perp}$ Signatures			IBE Encryption	
Curve	Key Gen	Sign	Verify	Key Gen	Sign	Verify	Encrypt	Decrypt
$SS(97,+)$	5	12	43	5	12	43	26	20
Curve A	3	74	95	65	6	90	50	37
$SS(163,-)$	13	14	106	13	14	106	101	76
Curve B	5	110	129	95	10	123	92	59
$SS(193,-)$	22	53	200	22	53	200	140	113
Curve C	8	167	189	141	15	184	117	82
$SS(239,-)$	30	48	258	30	48	258	190	156
Curve D	9	232	267	200	15	264	182	124
Curve E	19	435	461	193	20	457	306	203
$SS(353,-)$	78	93	730	78	93	730	583	463

References

1. S.S. Al-Riyami, J. Malone-Lee and N.P. Smart. Escrow-free encryption supporting cryptographic workflow. Preprint, 2004.
2. P.S.L.M. Barreto, H.Y. Kim, B. Lynn and M. Scott. Efficient algorithms for pairing-based cryptosystems. In *Advances in Cryptology – CRYPTO 2002*, Springer LNCS 2442, 354–369, 2002.
3. P.S.L.M. Barreto, B. Lynn and M. Scott. On the Selection of Pairing-Friendly Groups. In *Selected Areas in Cryptography (SAC)*, Springer-Verlag LNCS 3006, 17–25, 2004.
4. P.S.L.M. Barreto and M. Naehrig. Pairing-friendly elliptic curves of prime order. Preprint 2005.
5. D. Boneh and X. Boyen. Short signatures without random oracles. In *Advances in Cryptology – EUROCRYPT 2004*, Springer LNCS 3027, 56–73, 2001.

Table 4. Message sizes for Cryptographic Operations

Curve	BLS Signatures			BLS$^\perp$ Signatures			IBE Encryption	
	Key Size		Signature	Key Size		Signature	Private	
	Private	Public	Size	Private	Public	Size	Key	Ciphertext
Supersingular	m	m	m	m	m	m	m	$2m + l$
MNT	m	m	$3 \cdot m$	m	$3 \cdot m$	m	$3 \cdot m$	$2m + l$

6. D. Boneh, X. Boyen and H. Shacham. Short group signatures. In *Advances in Cryptology – CRYPTO 2004*, Springer LNCS 3152, 41–55, 2004.
7. D. Boneh and M. Franklin. Identity based encryption from the Weil pairing. In *Advances in Cryptology – CRYPTO 2001*, Springer LNCS 2139, 213–229, 2001.
8. 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.
9. D. Coppersmith. Evaluating logarithms in $GF(2^n)$. In *STOC 1984*, 201–207, 1983.
10. I. Duursma and H.-S. Lee. Tate pairing implementation for hyperelliptic curves $y^2 = x^p - x + d$. In *Advances in Cryptology – ASIACRYPT 2003*, Springer LNCS 2894, 111-222, 2003.
11. S. Galbraith, K. Harrison and S. Soldera. Implementing the Tate pairing. In *Algorithmic Number Theory Symposium – ANTS V*, Springer LNCS 2369, 324–337, 2002.
12. R. Granger, A. Holt, D. Page, N.P. Smart, F. Vercauteren. Function Field Sieve in Characteristic Three. In *Algorithmic Number Theory Symposium - ANTS VI*, Springer LNCS 3076, 223–234, 2004.
13. 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.
14. T. Izu and T. Takagi. Efficient computations of the Tate pairing for the large MOV degrees. In *International Conference on Information Security and Cryptology – ICISC 2002*, Springer LNCS 2587, 283–297, 2003.
15. A. Joux. A one round protocol for tripartite Diffie–Hellman. In *Algorithmic Number Theory Symposium – ANTS IV*, Springer LNCS 1838, 385–394, 2000.
16. A. Joux and R. Lercier. The function field sieve is quite special. In *Algorithmic Number Theory Symposium – ANTS V*, Springer LNCS 2369, 431–445, 2002.
17. R. Lercier. Discrete logarithms in $GF(p)$. Posting to NMBRTHRY List, 2001.
18. N. Li, W. Du and D. Boneh. Oblivious signature-based envelope. In *22nd ACM Symposium on Principles of Distributed Computing (PODC)*, 182–189, 2003.
19. A. Miyaji, M. Nakabayashi and S. Takano. New explicit conditions of elliptic curve traces for FR-reduction. In *IEICE Transactions on Fundamentals*, **E84-A** (5), 1234–1243, 2001.
20. V. Miller. Short programs for functions on curves. Unpublished manuscript, 1986.
21. J. Robertson. Solving the generalized Pell equation. Available at `http://hometown.aol.com/jpr2718/`.
22. R. Sakai, K. Ohgishi and M. Kasahara. Cryptosystems based on pairings. In *Proc. SCIS 2000*, 2000.
23. R. Sakai, K. Ohgishi and M. Kasahara. Cryptosystems based on pairings over elliptic curves. In *Proc. SCIS 2001*, 2001.

24. O. Schirokauer. Using number fields to compute logarithms in finite fields. *Math. Comp.*, **69**, 1267–1283, 2000.
25. M. Scott. Complex multiplication program. Available at `ftp.compapp.dcu.ie/pub/crypto/cm.exe`.
26. M. Scott and P.S.L.M. Barreto. Generating more MNT elliptic curves. In *Cryptology ePrint Archive*, Report 2004/058, 2004.
27. E. Thomé. Computation of discrete logarithms in $GF(2^{607})$. In *Advances in Cryptology – ASIACRYPT 2001*, Springer LNCS 2248, 107–124, 2001.

7 Appendix

Curve A – 160 bits

Field: $\mathbb{F}_q$ where q is given by

```
8C72D321E48AA1419B22F914CB43C112B76D7AE5
```

Curve: $y^2 = x^3 - 3x + b$, where b is given by

```
 299CE219B7B01348FC2B5007B6AB1EE1005676F7
```

Order: N where N is given by

```
8C72D321E48AA1419B23B6B2E4A85A073822640F
```

Cofactor: $h = 1$ **Extension Field:** We have $\alpha = 6$ and $\mathbb{F}_{q^\alpha}$ is defined by $\chi^6 + \chi^2 + 3 = 0$.

Curve B – 192 bits

Field: $\mathbb{F}_q$ where q is given by

```
BF52ED99D5808F126790D7DC18D901B076429F3A2FA78F65
```

Curve: $y^2 = x^3 - 3x + b$, where b is given by

```
 7EEAFAF4178E7349192E71FA4EB40C681A11A9B5B4F2C0C9
```

Order: N where N is given by

```
5FA976CCEAC0478933C86BEE93F2F8C16A54AE0A732FF4B5
```

Cofactor: $h = 2$ **Extension Field:** We have $\alpha = 6$ and $\mathbb{F}_{q^\alpha}$ is defined by $\chi^6 + 2 = 0$.

Curve C – 222 bits

Field: $\mathbb{F}_q$ where q is given by

```
20DF589D615A00DE349A7B4179B6BA507C693FF8ECC83614A610AAC3
```

Curve: $y^2 = x^3 - 3x + b$, where b is given by

```
0F99D400C2C7DED3542EAA3662E551B389489A8D38C69EE1A818753F
```

Order: N where N is given by

```
106FAC4EB0AD006F1A4D3DA0BCDB24FB28F7F39C248E644D4FD14077
```

Cofactor: $h = 2$ **Extension Field:** We have $\alpha = 6$ and $\mathbb{F}_{q^\alpha}$ is defined by $\chi^6 + 4 = 0$.

Curve D – 256 bits

Field: $\mathbb{F}_q$ where q is given by

```
F6529C2A424A6332B1D5054E2F7B68AAEE7EF91874DD140C6919AF9B71 \\
9ED905
```

Curve: $y^2 = x^3 - 3x + b$, where b is given by

```
6E974D68EF44F266AE3DD5D1F97C497C1D5452D1B074A6C06A25D4E581 \\
9CCD1C
```

Order: N where N is given by

```
F6529C2A424A6332B1D5054E2F7B68ABE99C585A8419AE9FB45C620E5E \\
F666C3
```

Cofactor: $h = 1$ **Extension Field:** We have $\alpha = 6$ and $\mathbb{F}_{q^\alpha}$ is defined by $\chi^6 + 6 = 0$.

Curve E – 307 bits

Field $\mathbb{F}_q$ where q is given by

```
05F9732C02629855B99FD12895E6BDBE0BB706EFA108E0C07AF66AAD00E \\
B1F0F5989C33BD1C4E5
```

Curve: $y^2 = x^3 - 3x + b$, where b is given by

```
05607CD7395B5F49C34A289E4072C37A56601B69C8F64F6BA3F827C87D \\
EE8279BC2E640F16C279
```

Order: N where N is given by

```
05F9732C02629855B99FD12895E6BDBE0BB706ED2F4DC3D3182475E37D3 \\
C9FA61B41FD46D6868F
```

Cofactor: $h = 1$ **Extension Field:** We have $\alpha = 6$ and $\mathbb{F}_{q^\alpha}$ is defined by $\chi^6 + 5 = 0$.