Faster Pairings using an Elliptic Curve with an Efficient Endomorphism

Michael Scott

School of Computing
Dublin City University
Ballymun, Dublin 9, Ireland.
mike@computing.dcu.ie

Abstract. The most significant pairing-based cryptographic protocol to be proposed so far is undoubtedly the Identity-Based Encryption (IBE) protocol of Boneh and Franklin. In their paper [6] they give details of how their scheme might be implemented in practise on certain supersingular elliptic curves of prime characteristic. They also point out that the scheme could as easily be implemented on certain special non-supersingular curves for the same level of security. An obvious question to be answered is – which is most efficient? Motivated by the work of Gallant, Lambert and Vanstone [12] we demonstrate that, perhaps counter to intuition, certain ordinary curves closely related to the supersingular curves originally recommended by Boneh and Franklin, provide better performance. We illustrate our technique by implementing the fastest pairing algorithm to date (on elliptic curves of prime characteristic) for contemporary levels of security. We also point out that many of the non-supersingular families of curves recently discovered and proposed for use in pairing-based cryptography can also benefit (to an extent) from the same technique.
Keywords: Tate pairing implementation, pairing-based cryptosystems.

1 Introduction

If it is to be successful in the long term, pairing-based cryptography needs efficient algorithms for the calculation of the Weil or Tate pairing. In his early text book Menezes [15] mentions an implementation of the Weil pairing which "reported running times of just a few minutes" on a SUN-2 SPARC-station. However this is more than a little unfair – at the time there was no real incentive to try and optimise the standard technique, based on Miller's algorithm [16]. The development of protocols that require fast pairings has produced a series of improvements and tricks which have drastically reduced this running time down to just a few milliseconds.

One target might be that the pairing calculation should take as long as an RSA decryption, for the same level of security, and as pointed out by Scott [18], this target has already almost been reached. However more improvements may be possible, and it is the purpose of this paper to illustrate a new method which

can either produce a further speed-up of up to 20%, or half the amount of storage required, depending on the context in which the pairing is to be calculated.

The development of fast pairings has advanced on two fronts. The first has concentrated on optimising algorithms for the Tate pairing on elliptic curves of prime characteristic, both supersingular and ordinary. The second has focused on algorithms for supersingular curves of small characteristic, typically of characteristic 2 and 3. The former approach is epitomised by the work of Barreto, Kim, Lynn and Scott [2] and Galbraith, Harris and Soldera [11]. For an easy-to-read description of the so-called BKLS-GHS algorithm, with timings, see [18]. While the BKLS-GHS algorithm is also suitable for use over small characteristic curves, the work of Duursma and Lee [10] made it clear that a more efficient algorithm was possible in this context. This approach culminated in the work of Barreto, Galbraith, O'hEigeartaigh and Scott [1], which introduced the primitive η_T pairing, and showed how the Tate pairing could be calculated from it using an iterative loop only half the size of that required by Duursma and Lee. They also raise the possibility that pairings over characteristic 2 supersingular hyperelliptic curves may also be competitive.

However comparing the two types of fast pairings is difficult, as it amounts to comparing the difficulty of the discrete logarithm problem in fields of prime characteristic, with that in fields of small characteristic. In our view this comparison has not been adequately experimentally investigated. However the most authoritative comparison that we have found is due to Lenstra [14]. From this, and using the timings from [1], it would appear that the η_T approach may in fact be the fastest. However since there is still some concern that the discrete logarithm problem in fields of low characteristic may be easier than we currently think, in this paper we will concentrate exclusively on the prime characteristic case.

It has been suggested that pairings might be speeded up by using a prime modulus p of low Hamming weight [13]. This idea can be used with both supersingular and non-supersingular curves. However this also raises legitimate concerns about a possible lowering of discrete-logarithm security. We will not consider the use of a prime modulus of low Hamming weight here.

A critical parameter of any pairing implementation is the embedding degree, or "security multiplier", denoted k. For reasons of efficiency it is usually recommended that k be even [2] . The security multiplier relates the size of the base field over which points on the elliptic curve are manipulated, with the discrete-logarithm security of the pairing. For example on a particular supersingular hyperelliptic curve of characteristic 2, a value of $k = 12$ is possible [1], and so a hyperelliptic curve over the field $\mathbb{F}_{2^{113}}$ would result in a pairing with the discrete logarithm security of a 12*113=1356 bit binary extension field, which by reference to [14] might be considered to be adequately secure. So a large security multiplier implies that we can work on an elliptic or hyperelliptic curves over a smaller base field, with efficiency advantages. However the advantage of a large security multiplier is perhaps not as great as one might think, as the major part

of the pairing calculation involves manipulations over the extension field (of size 1356 bits in our example), rather than over the smaller base field.

In the case of prime characteristic fields, the use of a security multiplier of $k = 2$ has proven to be surprisingly efficient for contemporary levels of security [18]. Note that for supersingular elliptic curves of prime characteristic, $k = 2$ is the maximum possible. The issue of how to scale security in pairing based protocols has been considered by both Koblitz and Menezes [13], and by Scott [19]. The consenus is that the appropriate way to scale security is to increase the security multiplier rather than increase the size of the prime modulus.

Note that by "contemporary levels of security", we mean a 1024-bit prime extension field size, and a group size of 160-bits. This implies roughly the same security as 1024-bit RSA [19].

Here we are concerned with the calculation of the Tate pairing, denoted $e(P, Q)$, which evaluates as an element of order r in $\mathbb{F}_{p^k}$ where P is a point of order r on $E(\mathbb{F}_p)$ and Q is a point on $E(\mathbb{F}_{p^k})$.

2 Supersingular Curves

In their original paper [6], Boneh and Franklin recommend the use of either of these supersingular curves over $\mathbb{F}_p$

$$y^2 = x^3 + Ax, \text{ where } p \equiv 3 \bmod 4 \tag{1}$$

$$y^2 = x^3 + B, \text{ where } p \equiv 2 \bmod 3 \tag{2}$$

On supersingular curves the modified pairing is calculated as $\hat{e}(P, Q) = e(P, \psi(Q))$, where $e(P, Q)$ denotes the Tate pairing, and $\psi(.)$ denotes the distortion map. For the first curve an appropriate distortion map is defined as $\psi_1 : (x, y) \to (-x, \alpha y)$ where $\alpha = \sqrt{-1}$, and for second curve the distortion map is $\psi_2 : (x, y) \to (\beta x, y)$ where β is a non-trivial cube root of unity. Note that both α and β are elements of the extension field $\mathbb{F}_{p^2}$, corresponding to the security multiplier value of $k = 2$.

In the Boneh and Franklin IBE scheme there is a necessity to hash identities to curve points. For the second curve this can be done by hashing the identity string to the y coordinate, and then solving the modular cubic equation for x. Since $p \equiv 2 \bmod 3$, this will always be possible. For the first curve one could hash the identity to x and test if $x^3 + ax$ is a quadratic residue. If it is not then negate x. Then solve the modular quadratic for y, and choose one of the two solutions according to some convention. This will always work as $p \equiv 3 \bmod 4$ implies that -1 is a quadratic non-residue.

3 Not Supersingular Curves

Consider now these non-supersingular curves over $\mathbb{F}_p$

$$y^2 = x^3 + Ax, \text{ where } p \equiv 1 \bmod 4 \tag{3}$$

$$y^2 = x^3 + B, \text{ where } p \equiv 1 \bmod 3 \tag{4}$$

Recall that IBE can equally well be implemented on these curves, using the Tate pairing $e(P, Q)$ directly. Suitable curves can be found with $k = 2$, but of course we are no longer restricted to this value alone – larger values of k are also possible (see below).

Note that all that has been changed is the congruence conditions applying to p. For our convenience here we will describe these curves as the not-supersingular (NSS) curves to distinguish them from the generality of ordinary curves. Under these circumstances what becomes of the distortion maps? Well of course they are no longer distortion maps, as now $\alpha, \beta \in F_p$. However these mappings continue to be useful, as we will see, not as distortion maps, but rather as efficient *endomorphisms*.

What about hashing identities to curve points on these curves? One interesting feature of $k = 2$ curves is that both the curve and its quadratic twist have the same embedding degree of $k = 2$. This arises from the condition [2] that the group order r divides both $p + 1$ and $p + 1 - t$ (the number of points on the curve), where t is the trace of the Frobenius for the particular curve. Therefore it follows that r also divides $p + 1 + t$, the number of points on the quadratic twist of the curve, and either curve is a suitable vehicle for IBE.

To be concrete we will from this point on concentrate our attention to the curve of equation (4), although all our results apply equally to the other curve. For simplicity choose $p = 7 \bmod 12$, so that -1 is a quadratic non-residue, and a quadratic twist of the original curve is given by $y^2 = x^3 - B$. Then if an identity is hashed to a value x, if $x^3 + B$ is not a quadratic residue, then $(-x)^3 - B$ will be. So the trick is to hash to either the original curve or to the twisted curve, depending on the identity. IBE public parameters for both the curve and its twist must be maintained, but other than that the IBE implementation will proceed smoothly with negligible additional overhead.

4 Curve generation

While generating suitable parameters for supersingular curves is easy, it is much more challenging to generate suitable non-supersingular curves. However a lot of progress has been made. Just as pairing-based cryptography was getting started, Miyaji, Nakabayashi and Takano [17] described a method for generating non-supersingular curves with embedding degrees of 3, 4 and 6. Barreto, Lynn and Scott [3] were the first to provide simple formulae for generating whole families of curves with useful embedding degrees. Their results were extended by Brezing and Weng [7], and later by Barreto and Naehrig [4] who came up with formulae which allow the generation of ideal $k = 12$ curves with many useful properties.

We will return to these curves later, but for now will just make the observation that the majority of such curves are of the not-supersingular form.

By far the most general method for generating pairing-friendly non-supersingular curves in that due to Cocks and Pinch [5], and this is the method that we shall use to generate $k = 2$ not-supersingular curves suitable as replacements for the standard supersingular curves described above, for use with Boneh and Franklin IBE. Crucially (for us) the Cocks-Pinch algorithm allows a free choice of the group order r. It is well known that a choice of a low Hamming weight r speeds up the Tate pairing calculation [2], [11], [18]. Alternatively it suffices if a small multiple of r has a low Hamming weight.

While these methods provide formulae for determining the prime modulus p and the trace of the Frobenius t of the desired curve, they do not generate the curve parameters directly. For this the method of Complex Multiplication must be used [9]. Note that the not-supersingular curves are associated with a CM discriminant of $D = -4$ and $D = -3$ respectively. In these cases determining the curve parameters is particularly simple, as for example demonstrated in [4].

For our NSS curve (4), the Cocks-Pinch algorithm can be described very simply: Select a suitable r of low Hamming weight (or a small multiple of which has low hamming weight). Calculate $v = \sqrt{-4/3} \bmod r$. Set $t = \omega r$. Keep adding r to v until $p = (3v^2 + t^2)/4$ is prime. (Choose ω so that p is 512 bits). Note that $r \mid p + 1 - t$. Finally use the CM method to find the curve parameter B associated with the curve of order $p + 1 - t$. (There are 6 possible group orders generated by the CM method in this case [9], so care must be taken to choose the right one).

5 Efficient pairings on NSS curves

In their paper Gallant, Lambert and Vanstone [12] describe an efficient method for point multiplication, that applies to NSS curves, and indeed this paper is the main source of inspiration for the current work, although we exploit the endomorphism in a completely different way. Another motivation comes from consideration of the η_T pairing as described in [1].

In the course of calculating the Tate pairing $e(P, Q)$ the first parameter, the point P, is multiplied by its order. Of course this results in the point-at-infinity. In the course of this process the intermediate values of this point, as it typically follows a simple double-and-add trajectory to its pre-ordained destination, along with the second parameter Q, are used to accumulate the pairing value. In some contexts the value of P may be fixed and known in advance (for example it may be a public parameter, or a private key). In this case these intermediate values can be precalculated and stored, with some performance benefit [18].

Choosing as a parameter a group order r of low Hamming weight drastically reduces the number of expensive add steps, and hence speeds the algorithm. However does this process take full advantage of our ability to choose r? The intuition that led to the discovery of the η_T pairing was that the multiplication by the group order could perhaps be divided into two parts. At the "half-way"

stage, the point P might be "close" to where it started. Therefore the second half of the iteration would hopefully be a simple function of the first part, and so only half the number of iterations might be required. Here we demonstrate that something similar can be achieved for NSS curves.

Gallant, Lambert and Vanstone [12] have pointed out the following useful facts about NSS curves, and the efficient endomorphisms that they support.

For the curve (3) let P be a point of prime order r, with coordinates (x, y), such that $\lambda^2 + 1 = 0 \bmod r$. Then the point λP has coordinates $(-x, \alpha y)$, where α is a square root of -1 mod p. Note that there are two possibilities for α, depending on the two possible solutions of the quadratic equation for λ. The endomorphism is defined as $\phi_1 : (x, y) \rightarrow (-x, \alpha y)$

For the curve (4) let P be a point of prime order r, with coordinates (x, y), such that $\lambda^2+\lambda+1 = 0 \bmod r$. Then the point λP has coordinates $(\beta x, y)$, where β is a non-trivial cube root of unity mod p. Note that there are two possibilities for β, depending on the two possible solutions of the quadratic equation for λ. The endomorphism is this case is defined as $\phi_2 : (x, y) \rightarrow (\beta x, y)$

This imples that given a point P on one of these curves, one can immediately determine a fixed multiple of the point, with a single field multiplication. In [12] this is exploited to develope a fast point multiplication algorithm.

Focusing on the latter curve, our idea is that λ should be chosen in advance, of low Hamming weight. For example we might choose $\lambda = 2^n$. Then select as r a large prime divisor of $\lambda^2+\lambda+1$. For example $n = 87$, and $r = (2^{174}+2^{87}+1)/73$ (a 168-bit prime) would seem to be a suitable choice. Using a double and add algorithm for the calculation of the Tate pairing would require the calculation of $2^n(2^nP + P) + P$ [1]. However using the endomorphism we can immediately know the value of $2^nP + P$. And this implies that we know the sequence of points that will occur during the final 2^n doublings, without having to explicitly calculate them (exploiting the well-known commutativity of the endomorphism with point multiplication: $a\phi(P) = \phi(aP)$). This results in significant computational savings.

Now we explain our technique in a little more detail. Given the point P with coordinates (x, y), and using the endomorphism, it is easy to calculate 2^nP+P as the point $(-(\beta + 1)x, -y)$. Clearly if the values of the initial 2^n point doublings were stored, the values of the final 2^n doublings can be found at the cost of a single field multiplication, resulting in a faster algorithm. Alternatively if the value of P is fixed, only half the storage would be required. Either way the result is a more efficient algorithm. However it is possible to do a little better than this.

For the first n iterations of Miller's algorithm the contribution to the pairing value is $(y_Q - y_i) - m_i(x_Q - x_i)$, where (x_i, y_i) is the point 2^iP, m_i is the line slope resulting from the current point doubling, and (x_Q, y_Q) is the point Q. This value can be multiplied at will by any element of $\mathbb{F}_p$, as the effect of any such multiplication will be wiped out by the final exponentiation. For the final n iterations the contribution will be $(y_Q + y_i) + (\beta^2 + 1)m_i(x_Q + (\beta + 1)x_i)$ –

[1] As pointed out by Duursma and Lee [10] the final point addition can be omitted without changing the value of the pairing.

note the adjusted value of the slope. But since β is a non-trivial cube root of unity, we know that $\beta^2+\beta+1 = 0 \bmod p$. Substituting and simplifying we have $(y_Q+y_i)+(\beta^2+1)m_i(x_Q+(\beta+1)x_i) = (y_Q+y_i)+m_i(\beta x_Q - x_i)$. Now multiply by -1 to obtain the equivalent contribution of $(-y_Q - y_i) - m_i(\beta x_Q - x_i)$.

Observe therefore that we can obtain the same values by switching the point Q to $\bar{Q}$ for the final n iterations, where $\bar{Q} = (\beta x_Q, -y_Q)$, and using exactly the same sequence of (x_i, y_i) as we did for the first n iterations.

5.1 A basic algorithm

We are now ready to bring these ideas together and describe our modified BKLS-GHS algorithm in detail. First we equip ourselves with a library which can add or double points on an elliptic curves by means of a function $A.add(B)$ which adds B to A, and returns the line slope m.

Next we need a function $g(.)$ to calculate the contribution of the current iteration to the pairing value. The returned value is used for the first n iterations, and the values for the final n iterations are calculated at the same time (at very little extra cost) and stored for later use.

Algorithm 1 Function $g(.)$

INPUT: A, B, Q, i
1: $x_i, y_i \leftarrow A$
2: $x_Q, y_Q \leftarrow Q$
3: $m_i = A.add(B)$
4: **store** $-y_Q - y_i - m_i(\beta x_Q - x_i)$ in an array element $s[i]$
5: **return** $y_Q - y_i - m_i(x_Q - x_i)$

In practise the function $A.add(B)$ will be faster if the point A is represented in projective coordinates, which makes for a somewhat more complex $g(.)$ function. We omit the details, except to point out that much of the calculation is shared between the stored point and the returned point, with further savings.

For optimal performance the point Q is deliberately placed into the trace-zero subgroup, which means that only the variable y_Q is in $\mathbb{F}_{p^2}$. The variables x_i, y_i, x_Q, m_i are all in $\mathbb{F}_p$. See [18] for details. The returned and stored values are in $\mathbb{F}_{p^2}$.

Care must be taken to ensure that correct non-trival cube root of unity for β is chosen, as there are two possibilities associated with the two solutions for $\lambda^2 + \lambda + 1 = 0 \bmod r$. The right value can easily be found by trial and error, and the value of βx_Q can then be precalculated and stored.

For the particular case $n = 87$ the full Tate pairing algorithm is given in Algorithm 2. This algorithm will also work for any choice of $\lambda = 2^n$ which leads to a near-prime value of $r = \lambda^2 + \lambda + 1$. However in practise, and in the range of useful values, good values for n are hard to find.

Algorithm 2 Computation of $e(P,Q)$ on NSS curve (4), $k=2$, $\lambda = 2^{87}$, $r = (\lambda^2+\lambda+1)/73$

INPUT: P, Q
OUTPUT: $e(P,Q)$
1: $A \leftarrow P$, $f \leftarrow 1$
2: **for** $i \leftarrow 1$ **to** 87 **do**
3: $\quad f \leftarrow f^2.g(A,A,Q,i)$
4: **end for**
5: $f \leftarrow f.g(A,P,Q,-)$
6: **for** $i \leftarrow 1$ **to** 87 **do**
7: $\quad f \leftarrow f^2.s[i]$
8: **end for**
9: **return** $f^{(p-1)(p+1)/r}$

5.2 A better algorithm

Consider now the slightly more complicated choice of $\lambda = 2^a + 2^b$. In this case we have much greater control of r, and it is much easier to find a prime value which still has a very low Hamming weight. For example choosing $\lambda = 2^{80} + 2^{16}$ gives a prime $r = \lambda^2 + \lambda + 1$ of 161 bits. A slight complication arises in this case, as the multiplication of the point P by r no longer follows a purely double-and-add algorithm, and so Miller's algorithm needs to be slightly modified to accomodate this. In the following algorithm 3, the variable h is used to handle this modification.

The extra storage requirement for the array s is not very large, but in some circumstances it may become an issue. An alternative version of the algorithm requires no storage, for a little extra work. See algorithm 4.

6 Results

An important first step in an implementation is to use the Cocks and Pinch algorithm to generate a suitable 512-bit, k=2 NSS elliptic curve. The curve $y^2 = x^3 + 5 \bmod p$, for the 512-bit prime p, where

$$p = 11457475683995493806353174186205825314535461236767597441115533728505070527823$$
$$15453265765699123447398664170319394034355982362866887873432690950208939349364 3$$

was quickly found. The Tate pairing was calculated on this curve using algorithms 3 and 4 with $a = 80$ and $b = 16$, and compared with the pairing calculated on the original Boneh and Franklin supersingular curve. As anticipated, the NSS curve pairings are significantly faster.

We provide both timings and a count of the total number of $\mathbb{F}_p$ modular multiplications and squarings required. In the implementation we used a final exponentiation based on the calculation of a Lucas sequence, which allows easy times-two compression of the output, as described in [18] and [20].

Algorithm 3 Computation of $e(P,Q)$ on NSS curve (4), $k=2$, $\lambda = 2^a + 2^b$, $a > b$, $r = \lambda^2 + \lambda + 1$. Requires an array s of length a.

INPUT: P, Q
OUTPUT: $e(P,Q)$

$A \leftarrow P$, $f \leftarrow 1$, $j \leftarrow 1$
for $i \leftarrow 1$ **to** $a - b$ **do**
    $f \leftarrow f^2.g(A, A, Q, j{+}{+})$
end for
$f \leftarrow f.g(A, P, Q, j{+}{+})$
for $i \leftarrow 1$ **to** b **do**
    $f \leftarrow f^2.g(A, A, Q, j{+}{+})$
end for
$f \leftarrow f.g(A, P, Q, -)$
$h \leftarrow f$, $j \leftarrow 1$
for $i \leftarrow 1$ **to** $a - b$ **do**
    $f \leftarrow f^2.s[j{+}{+}]$
end for
$f \leftarrow f.s[j{+}{+}]$
$f \leftarrow f.h$
for $i \leftarrow 1$ **to** b **do**
    $f \leftarrow f^2.s[j{+}{+}]$
end for
return $f^{(p-1)(p+1)/r}$

Algorithm 4 Computation of $e(P,Q)$ on NSS curve (4), $k=2$, $\lambda = 2^a + 2^b$, $a > b$, $r = \lambda^2 + \lambda + 1$. No extra storage requirement.

INPUT: P, Q
OUTPUT: $e(P,Q)$

$A \leftarrow P$, $f_1 \leftarrow 1$, $f_2 \leftarrow 1$, $j \leftarrow 1$
for $i \leftarrow 1$ **to** $a - b$ **do**
    $f_1 \leftarrow f_1^2.g(A, A, Q, 0)$
    $f_2 \leftarrow f_2^2.s[0]$
end for
$f_1 \leftarrow f_1.g(A, P, Q, 0)$
$f_2 \leftarrow f_2.s[0]$
for $i \leftarrow 1$ **to** b **do**
    $f_1 \leftarrow f_1^2.g(A, A, Q, 0)$
    $f_2 \leftarrow f_2^2.s[0]$
end for
$f_1 \leftarrow f_1.g(A, P, Q, -)$
$f \leftarrow f_1^{\lambda}.f_2$
return $f^{(p-1)(p+1)/r}$

Table 1. NSS vs Supersingular Tate Pairing – 3GHz Intel PIV.

Curve type	$\mathbb{F}_p$ muls	Time (ms)
512-bit, $k = 2$ Supersingular curve	4070	8.9
512-bit, $k = 2$ NSS curve, with storage	3163	7.2
512-bit, $k = 2$ NSS curve, no storage	3329	7.5

7 Extensions

The basic idea can be extended in a few directions. Firstly the same basic technique will also work for the "other" NSS curve of equation (3). The idea could also be applied to the non-supersingular curves with CM discriminants of $D = -7$ and $D = -8$, as described in [12], although in these cases the savings would be much less significant as the endomorphisms are much more complex to calculate. In fact it would be more correct to refer to the class of exploitable curves as "small discriminant CM curves".

If the first parameter P is fixed, then no savings will be realised in terms of computation using this method as all the multiples of P can be precalculated and stored. However using NSS curves only half the storage will be required, which leads to greater efficiency, and which might be significant in a constrained environment.

The method has been described in the context of a security multiplier of $k = 2$, but it also applies immediately to higher values of k. In these cases the computational time savings will be smaller, as the implicit point multiplication of P by r becomes a less significant part of the overall calculation [19].

We note in passing that the method of Gallant, Lambert and Vanstone [12] also has direct application to pairing-based protocols which require point multiplication, such as the Boneh and Franklin IBE, if they are implemented on NSS curves. With our choice of group order the deployment of this scheme becomes particularly simple: Calculate kP as $k_1P + k_2\phi(P$, where $k_2 = k/\lambda, k_1 = k \bmod \lambda$. See [12] for details..

Finally we point out that many curves that have been suggested as suitable for use in pairing-based cryptography are in fact already of the NSS form [3], [7], and furthermore have a group order of the required form. For example the $k = 12$ curve suggested in Appendix A of [3] is of this form, as is the nice $k = 8$ curve suggested by Brezing and Weng [7] and implemented in [19]. This is facilitated by the group order of these curves being derived from a cyclotomic polynomial which, as luck would have it, is of the same form as $r = \lambda^2 + \lambda + 1$.

Are NSS curves any less secure than supersingular curves or indeed general pairing-friendly non-supersingular curves? There is no reason to think so. In standard elliptic curve cryptography some classes of curves are considered as weaker than others, although sometimes the reasons are not well supported by any hard evidence. For example it is considered in certain quarters (for example the German National Security Agency), that curves with smaller CM discrim-

inants are in some sense weaker than others [8]. Whatever the merits of such judgements, they do not apply in the pairing context, where it is already known that there is an index calculus attack on the extension field $\mathbb{F}_{p^k}$, which arises as a direct consequence of having a small embedding degree k. Of course by choosing p and k wisely this index calculus attack becomes infeasible. It remains an interesting open question whether or not there are any weak classes of pairing-suitable curves, supersingular or non-supersingular.

8 Conclusions

We have described a method of calculating pairings on certain non-supersingular curves, which is faster than using the equivalent supersingular curve, as originally recommended by Boneh and Franklin [6] for use in Identity Based Encryption. The proposed method is more efficient in terms of time or of space than any other method so far proposed for prime characteristic fields and at contemporary levels of security.

9 Acknowledgement

Thanks to Steven Galbraith for his comments on an early draft of this paper.

References

1. Paulo S. L. M. Barreto, Steven Galbraith, Colm O hEigeartaigh, and Michael Scott. Efficient pairing computation on supersingular abelian varieties. Cryptology ePrint Archive, Report 2004/375, 2004. `http://eprint.iacr.org/`.
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*, volume 2442 of *Lecture Notes in Computer Science*, pages 354–68. Springer-Verlag, 2002.
3. P.S.L.M. Barreto, B. Lynn, and M. Scott. Constructing elliptic curves with prescribed embedding degrees. In *Security in Communication Networks – SCN'2002*, volume 2576 of *Lecture Notes in Computer Science*, pages 263–273. Springer-Verlag, 2002.
4. P.S.L.M. Barreto and M. Naehrig. Pairing-friendly elliptic curves of prime order. Cryptology ePrint Archive, Report 2005/133, 2005. `http://eprint.iacr.org/`.
5. I. F. Blake, G. Seroussi, and N. P. Smart, editors. *Advances in Elliptic Curve Cryptography, Volume 2*. Cambridge University Press, 2005.
6. D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. *SIAM Journal of Computing*, 32(3):586–615, 2003.
7. F. Brezing and A. Weng. Elliptic curves suitable for pairing based cryptography. Cryptology ePrint Archive, Report 2003/143, 2003. Available from `http://eprint.iacr.org/2003/143`.
8. Johannes Buchmann and Harald Baier. Efficient construction of cryptographically strong elliptic curves. In *INDOCRYPT*, pages 191–202, 2000.
9. R. Crandall and C. Pomerance. *Prime Numbers: a Computational Perspective*. Springer-Verlag, Berlin, 2001.

10. I. Duursma and H. S. Lee. Tate-pairing implementations for tripartite key agreement. In *Advances in Cryptology – Asiacrypt 2003*, volume 2894 of *Lecture Notes in Computer Science*, pages 111–123. Springer-Verlag, 2003.
11. S. Galbraith, K. Harrison, and D. Soldera. Implementing the Tate pairing. In *Algorithm Number Theory Symposium – ANTS V*, volume 2369 of *Lecture Notes in Computer Science*, pages 324–337. Springer-Verlag, 2002.
12. R.P. Gallant, R.J. Lambert, and S.A. Vanstone. Faster point multiplication on elliptic curves with efficient endomorphisms. In *Advances in Cryptology – Crypto 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 190–200. Springer-Verlag, 2001.
13. Neal Koblitz and Alfred Menezes. Pairing-based cryptography at high security levels. Cryptology ePrint Archive, Report 2005/076, 2005. `http://eprint.iacr.org/`.
14. Arjen K. Lenstra. Unbelievable security. Matching AES security using public key systems. In *Advances in Cryptology – Asiacrypt 2001*, volume 2248, pages 67–86. Springer-Verlag, 2001.
15. A. Menezes. *Elliptic Curve Public Key Cryptosystems*. Kluwer Academic Publishers, 1993.
16. V. Miller. Short programs for functions on curves. unpublished manuscript, 1986.
17. A. Miyaji, M. Nakabayashi, and S. Takano. New explicit conditions of elliptic curve traces for FR-reduction. *IEICE Transactions on Fundamentals*, E84-A(5):1234–1243, 2001.
18. M. Scott. Computing the Tate pairing. In *CT-RSA*, volume 3376 of *Lecture Notes in Computer Science*, pages 293–304. Springer-Verlag, 2005.
19. M. Scott. Scaling security in pairing-based protocols. Cryptology ePrint Archive, Report 2005/139, 2005. `http://eprint.iacr.org/`.
20. M. Scott and P. Barreto. Compressed pairings. In *Advances in Cryptology – Crypto' 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 140–156. Springer-Verlag, 2004. Also available from `http://eprint.iacr.org/2004/032/`.