High Security Pairing-Based Cryptography Revisited

R. Granger, D. Page, and N.P. Smart

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

Abstract. The security and performance of pairing based cryptography has provoked a large volume of research, in part because of the exciting new cryptographic schemes that it underpins. We re-examine how one should implement pairings over ordinary elliptic curves for various practical levels of security. We conclude, contrary to prior work, that the Tate pairing is more efficient than the Weil pairing for all such security levels. This is achieved by using efficient exponentiation techniques in the cyclotomic subgroup backed by efficient squaring routines within the same subgroup. [1]

1 Introduction

In commercial cryptographic software libraries one typically employs Occam's Razor in limiting the number of implemented primitives and schemes to a minimum. The advantages of this approach are threefold: it reduces the programming, maintainence and security validation workload; it enables one to specialise and hence highly optimise the core operations; and it reduces the library footprint and usage of system resources. Around the time it was first proposed, one of the main criticisms levelled at standard elliptic curve cryptography was that there were too many options; it was hard for non-experts to decide on and construct the types of field and curve needed to satisfy performance and security constraints. Two decades later, pairing based cryptography is in a similar state in the sense that there are a huge range of parameterisation options, algorithmic choices and subtle trade-offs between the two. Hence, there is a real need to focus on a family of parameters which are flexible but offer efficient arithmetic and allow one to focus on a limited number of cases.

[1] The work described in this paper has been supported in part by the European Commission through the IST Programme under Contract IST-2002-507932 ECRYPT. The information in this document reflects only the author's views, is provided as is and no guarantee or warranty is given that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability

Generally speaking, a pairing is a non-degenerate bilinear map

$$t : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T.$$

Here we assume this pairing takes the concrete form

$$\hat{t} : E(\mathbb{F}_p) \times \overline{E}(\mathbb{F}_{p^{k/2}}) \longrightarrow \mathbb{F}_{p^k}^{\times}$$

where $\overline{E}$ is the quadratic twist of an elliptic curve E defined over $\mathbb{F}_{p^{k/2}}$. We restrict our attention to the case of ordinary elliptic curves and assume that $\#E(\mathbb{F}_p)$ is divisible by a large prime n which also divides $p^k - 1$ i.e., n is the order of the subgroups on which the pairing based protocols will be based. We let the respective subgroups of order n of the three groups involved be denoted $\mathbb{G}_1\mathbb{G}_2$ and $\mathbb{G}_T$ as is common in various papers on the subject.

Koblitz and Menezes [10] introduced the concept of pairing friendly fields. These are Kummer extensions of $\mathbb{F}_p$ defined by the polynomial

$$f(X) = X^k + f_0$$

for a values of $p \equiv 1 \pmod{12}$ and $k = 2^i 3^j$. Generally one assumes that k is even, which aids in efficiency due to the well known denominator elimination trick. Following [10] we particularly focus on the cases $k = 6$, 12 and 24. We let $f(\theta) = 0$ and define $\mathbb{F}_{p^k} = \mathbb{F}_p[\theta]$. Many protocols based on pairings perform arithmetic in the cyclotomic subgroup of $\mathbb{F}_{p^k}^{\times}$, which is the subgroup of order $\Phi_k(p)$. We denote this subgroup by $G_{\Phi_k(p)}$; the group $\mathbb{G}_T$ in the pairing above is contained in $G_{\Phi_k(p)}$. Hence if one is to implement pairing based protocols efficiently with such fields then one needs to be able to implement arithmetic efficiently.

The conclusion of [10] is that for high security levels the Weil pairing is to be preferred over the Tate pairing. The main result of this paper is that by optimising the exponentiation method used in the Tate pairing calculation one can in fact conclude the opposite: that in all cases the Tate pairing is the more efficient algorithm for all practical security levels.

In addition, we also look at efficient arithmetic in the group $G_{\Phi_k(p)}$ which will speed up both the Tate pairing and various protocols. This is inspired by work of Lenstra and Stam [14, 15] who introduce such efficient arithmetic in a specific family of finite fields of degree six, which are different from the pairing friendly fields. In particular, by restricting to $k = 6$ Lenstra and Stam present algorithms for arithmetic in the cyclotomic extension of $\mathbb{F}_p$ defined by the polynomial

$$g(X) = X^6 + X^3 + 1$$

when $p \equiv 2$ or $5 \pmod{9}$. We shall call such constructions cyclotomic fields of degree 6 in this paper. Lenstra and Stam present efficient squaring routines both for the finite field $\mathbb{F}_{p^6}$ and for the cyclotomic subgroup $G_{\Phi_6(p)}$ of order $\Phi_6(p)$, again $\mathbb{G}_T$ is contained in $G_{\Phi_6(p)}$. We let $g(\zeta) = 0$ and define $\mathbb{F}_{p^6}$, in this case, by $\mathbb{F}_p[\zeta]$. We shall describe how the use of cyclotomic fields, as opposed to the pairing friendly fields, can provide more efficient pairing algorithms when $k = 6$. We present an analogue of these results for pairing friendly fields which provides some efficiency improvement, but not as much as that achieved by Lenstra and Stam for cyclotomic fields of degree six. We leave it as an open research problem to generalise the results of Lenstra and Stam to cyclotomic fields of degree different from six. The only generalisation known is for fields of degree $6 \cdot 5^m$ [8], for which Lenstra and Stam's technique trivially applies.

The paper is organised as follows. In Section 2 we recap on the most efficient field arithmetic known for the two cases of finite fields mentioned above. In Section 3 we briefly recap on some standard formulae for the cost of elliptic curve operations. In Section 4 we recap on the model for estimating the cost pairings which was proposed by Koblitz and Menezes. Then in Section 5 we detail the implications of this model for our choice of finite fields.

2 Finite Field Operations

We let $m, \overline{M}, M$ (resp. $s, \overline{S}, S$) denote the time for multiplication (resp. squaring) in the fields $\mathbb{F}_p, \mathbb{F}_{p^{k/2}}$ and $\mathbb{F}_{p^k}$. In our analysis we shall assume that addition operations are cheap, however in a practical implementation for certain bit sizes the operation counts and algorithm choices we give may not be optimal due to this simplifying assumption.

We first note that if one is computing products (resp. squares) of polynomials of degree $2^i 3^j - 1$ over $\mathbb{F}_p$ then using the Karatsuba and Toom-Cook methods for multiplication and squaring this requires $v(k)$ multiplications (resp. squarings) in the field $\mathbb{F}_p$, where $v(k) = 3^i 5^j$.

2.1 Pairing Friendly Fields

As before we let $k = 2^i 3^j \geq 6$, let p denote a prime congruent to 1 modulo 12 and modulo k and define $\mathbb{F}_{p^k}$ via the polynomial $f(X) = X^k + f_0$. We assume throughout that f_0 has been chosen so that multiplication by f_0 can be performed quickly by simple additions rather than a full multiplication. Arithmetic in the subfield $\mathbb{F}_{p^{k/2}}$ is performed using the polynomial $X^{k/2} + f_0$, and mapping between the two representations is relatively straightforward.

The best algorithms for multiplication and squaring in $\mathbb{F}_{p^k}$ and $\mathbb{F}_{p^{k/2}}$ are the standard ones based on Karatsuba and Toom-Cook. Hence, in this case we obtain

$$\overline{M} = M/3 \quad \overline{S} = S/3$$

and

$$M \approx v(k)m \quad S \approx v(k)s$$

where $v(k) = 3^i 5^j$.

Inversion in the field $\mathbb{F}_{p^k}$ is computed by reduction to inversion in the subfield $\mathbb{F}_{p^{k/2}}$. If we let $\alpha = \sum_{i=0}^{k-1} a_i \theta^i \in \mathbb{F}_{p^k}$ then we can write

$$\alpha = \alpha_0 + \alpha_1 \theta$$

where $\alpha_0, \alpha_1 \in \mathbb{F}_{p^{k/2}}$ and are given by

$$\alpha_0 = \sum_{i=0}^{k/2-1} a_{2i}\theta^{2i} \text{ and } \alpha_1 = \sum_{i=0}^{k/2-1} a_{2i+1}\theta^{2i}.$$

We can thus compute

$$\Delta = \alpha_0^2 - \theta^2\alpha_1^2,$$

and

$$\alpha^{-1} = \frac{\alpha_0 - \alpha_1\theta}{\Delta}.$$

Inversion in $\mathbb{F}_{p^k}$ is therefore accomplished using two squarings, one inversion, and two multiplications in $\mathbb{F}_{p^{k/2}}$. Similarly, using the same idea one can reduce inversion in a cubic extension to three squarings, eleven multiplications and one inversion in the base field [9]. Iterating down through the subfields, for pairing-friendly fields inversion can thus be performed with just one inversion in $\mathbb{F}_p$, and a handful of multiplications. We summarize these costs, for the extensions which will interest us,

$$\begin{array}{ll} I_2 = 2s + 2m + \iota, & I_3 = 3s + 11m + \iota, \\ I_4 = 8s + 8m + \iota, & I_6 = 13s + 35m + \iota, \\ I_8 = 26s + 26m + \iota, & I_{12} = 43s + 51m + \iota, \\ I_{24} = 133s + 141m + \iota. & \end{array}$$

where I_t denotes the cost of inversion in $\mathbb{F}_{p^t}$ and ι denotes the cost of inversion in $\mathbb{F}_p$.

The Frobenius operation in pairing friendly fields is also efficiently computed as follows. If we define $\mathbb{F}_{p^k} = \mathbb{F}_p[\theta]/(f(\theta))$ then the Frobenius operation on the polynomial generator θ can be easily determined via

$$\theta^p = \theta^{k(p-1)/k+1} = (-f_0)^{(p-1)/k}\theta.$$

For later use we let $g = (-f_0)^{(p-1)/k} \in \mathbb{F}_p$ hence $\theta^p = g \cdot \theta$. Also now note that powers of the Frobenius operation are also easy to compute via

$$\theta^{p^i} = g^i \cdot \theta.$$

We also note that since k is even and $-f_0$ is a quadratic non-residue that we have

$$g^{k/2} = (-f_0)^{(p-1)/2} = -1.$$

In summary we conclude the operation counts for the various cases are as follows:

	$\mathbb{F}_{p^{k/2}}$		$\mathbb{F}_{p^k}$	
k	Mul	Sqr	Mul	Sqr
6	$5m$	$5s$	$15m$	$15s$
12	$15m$	$15s$	$45m$	$45s$
24	$45m$	$45s$	$135m$	$135s$

We now turn to the case of arithmetic in the subgroup $G_{\Phi_k(p)}$. For this subgroup we have that inversion comes for free. Let $\alpha \in G_{\Phi_k(p)}$, then since $\Phi_k(p)$ divides $p^{k/2}+1$ we have that

$$\alpha^{-1} = \alpha^{p^{k/2}}.$$

This leads to an inversion operation which can be performed using only $k/2$ negations in $\mathbb{F}_{p^k}$.

We can also improve the performance of squaring in this subgroup using a trick originally proposed by Lenstra and Stam [14, 15] in the context of finite extension fields defined by cyclotomic polynomials of degree 6. We first define

$$\alpha = \sum_{i=0}^{k-1} a_i \theta^i$$

where we now think of the coefficients a_i as variables. We then compute symbolically $\alpha^{p^{k/3}}$ and $\alpha^{p^{k/6}}$. One can then derive a set of equations defining the elements of the group $G_{\Phi_k(p)}$ via

$$\alpha^{p^{k/3}} \cdot \alpha - \alpha^{p^{k/6}} = \sum_{i=0}^{k-1} v_i \theta^i.$$

The variety defined by $v_0 = v_1 = \cdots = v_{k-1} = 0$ defines the set of elements of $G_{\Phi_k(p)}$. This follows since

$$\Phi_t(X) = X^{k/3} - X^{k/6} + 1$$

for all values of k arising in pairing friendly fields. As an example for the case $k = 6$ we obtain the set of equations

$$\begin{aligned}
v_0 &= -a_0 + {a_0}^2 + f_0 a_5 a_1 - f_0 {a_3}^2 + f_0 a_2 a_4, \\
v_1 &= g \cdot (-a_1 + 2 f_0 a_5 a_2 - f_0 a_3 a_4 + a_0 a_1), \\
v_2 &= (1-g) \cdot \left(a_2 - {a_1}^2 + a_0 a_2 - f_0 a_5 a_3 + f_0 {a_4}^2\right), \\
v_3 &= a_3 + 2 a_0 a_3 - a_2 a_1 + f_0 a_5 a_4, \\
v_4 &= g \cdot \left(a_0 a_4 + f_0 {a_5}^2 + a_3 a_1 - {a_2}^2 + a_4\right), \\
v_5 &= (1-g) \cdot (-a_5 + a_0 a_5 - 2 a_4 a_1 + a_3 a_2).
\end{aligned}$$

Note that for any $k \times k$ matrix Γ that

$$\alpha^2 = \alpha^2 + b \cdot (\Gamma \cdot v^t),$$

where $b = (1, \theta, \theta^2, \ldots, \theta^{k-1})$ and $v = (v_0, v_1, \ldots, v_{k-1})$. Hence, to find different forms of the squaring operation we simply need to select a matrix Γ which produces equations for squaring which are efficient.

A choice for Γ which seems to work well for $k = 6, 12$ and 24 is to set $\Gamma = \mathrm{diag}(d_1, d_2, d_3, d_1, d_2, d_3, \ldots, d_1, d_2, d_3)$ where

$$d_1 = 2, \quad d_2 = 2g^{k/6} - 2, \quad d_3 = -2g^{k/6}.$$

In this case for $k = 6$ we obtain the following formulae for squaring, if $\beta = \sum_{i=0}^{5} b_i\theta^i = \alpha^2$,

$$\begin{aligned}
b_0 &= -3\,f_0 {a_3}^2 + 3\,{a_0}^2 - 2\,a_0,\\
b_1 &= -6\,f_0 a_5 a_2 + 2\,a_1,\\
b_2 &= -3\,f_0 {a_4}^2 + 3\,{a_1}^2 - 2\,a_2,\\
b_3 &= 6\,a_0 a_3 + 2\,a_3,\\
b_4 &= 3\,{a_2}^2 - 3\,f_0 {a_5}^2 - 2\,a_4,\\
b_5 &= 6\,a_4 a_1 + 2\,a_5.
\end{aligned}$$

The formulae for $k = 12$ and $k = 24$ can be found in the Appendix.

Ignoring multiplication by f_0 and by small constants we then derive the following table detailing the comparative cost of squaring in both $\mathbb{F}_{p^k}$ and the subgroup $G_{\Phi_k(p)}$.

k	$\mathbb{F}_{p^k}$	$G_{\Phi_k(p)}$
6	$15s$	$6s + 3m$
12	$45s$	$12s + 18m$
24	$135s$	$24s + 84m$

Hence, we see that we have a significant improvement in the squaring operation for the subgroup $G_{\Phi_k(p)}$ although this improvement decreases as k increases.

2.2 Cyclotomic Fields of Degree 6

We recap on the techniques of [14, 15] for the finite fields $\mathbb{F}_p[\zeta]$, with $p \equiv 2 \pmod 9$. Elements in $\mathbb{F}_{p^6}$ are represented in the basis $\{\zeta, \zeta^2, \zeta^3, \zeta^4, \zeta^5, \zeta^6\}$. Using this representation multiplication in $\mathbb{F}_{p^6}$ can be performed using 15 multiplications in $\mathbb{F}_p$ (note that [14] gives the figure as 18 multiplications as the paper only considers Karatsuba and not Toom-Cook multiplication).

Squaring can be performed more efficiently using the fact that if we write $\alpha = \alpha_0\zeta + \alpha_1\zeta_4$, where α_i are polynomials in ζ of degree at most two, then one has

$$\alpha^2 = (\alpha_0 - \alpha_1)(\alpha_0 + \alpha_1)\zeta^2 + (2\alpha_0 - \alpha_1)\alpha_1\zeta^5.$$

Since, the α_i are of degree at most two this above formulae requires 10 multiplication in $\mathbb{F}_p$ to perform a squaring operation in $\mathbb{F}_{p^6}$.

Arithmetic in the subfield $\mathbb{F}_{p^3}$ is performed as in [9]. We set $\psi = \zeta + \zeta^{-1}$ and define $\mathbb{F}_{p^3} = \mathbb{F}_p[\psi]$. As a basis for $\mathbb{F}_{p^3}$ we take $\{1, \psi, \psi^2 - 2\}$. Via Toom-Cook multiplication (resp. squaring) requires 5 multiplications (resp. squares) in $\mathbb{F}_p$. As noted in Section 2.1 inversion in $\mathbb{F}_{p^3}$ can be performed in 11 multiplications in $\mathbb{F}_p$ and one inversion in $\mathbb{F}_p$.

Using this subfield inversion an inversion operation can be defined for $\mathbb{F}_{p^6}$.This inversion is carried out, in the language of [9], by mapping our $\mathbb{F}_{p^6}$ element to the representation F_2 and then performing the inversion in that representation and then mapping back to our representation. The conversion between representations requires four $\mathbb{F}_p$ multiplications, whilst the inversion in the F_2 representation requires $4\overline{S}$ plus application of the inversion in $\mathbb{F}_{p^3}$. Hence, requiring a total of 26 multiplications in $\mathbb{F}_p$ and one inversion in $\mathbb{F}_p$.

We now turn to the subgroup $G_{\Phi_6(p)}$. As before, inversion comes for free via the operation of the Frobenius map. Multiplication is performed just as for the full finite field, however squaring can be performed significantly faster using the equations contained in [14, 15]. If we let $\alpha = \sum_{i=0}^{5} a_i\zeta^{i+1} \in G_{\Phi_6(p)}$ and set $\beta = \sum_{i=0}^{5} b_i\zeta^{i+1} = \alpha^2$ then we have

$$\begin{aligned}
b_0 &= 2a_1 + 3a_4(a_4 - 2a_1),\\
b_1 &= 2a_0 + 3(a_0 + a_3)(a_0 - a_3),\\
b_2 &= -2a_5 + 3a_5(a_5 - 2a_2),\\
b_3 &= 2(a_1 - a_4) + 3a_1(a_1 - 2a_4),\\
b_4 &= 2(a_0 - a_3) + 3a_3(2a_0 - a_3),\\
b_5 &= -2a_2 + 3a_2(a_2 - 2a_5).
\end{aligned}$$

Hence, squaring requires six $\mathbb{F}_p$ multiplications. The operation counts for the various cases are as summarised by the following table:

$\mathbb{F}_{p^3}$		$\mathbb{F}_{p^6}$		$G_{\Phi_6(p)}$	
Mul	Sqr	Mul	Sqr	Mul	Sqr
$5m$	$5m$	$15m$	$10m$	$15m$	$6m$

2.3 Exponentiation in $G_{\Phi_k(p)}$

Finally, we address the issue of exponentiation, by an exponent e, of elements in the cyclotomic subgroup $G_{\Phi_k(p)}$ of $\mathbb{F}_{p^k}^{\times}$ which has order divisible by n. Using Lucas sequences [13] this can be accomplished in time

$$C_{\text{Luc}}(e) = (\overline{M} + \overline{S}) \log_2 e.$$

However, one could also use exponentiation via standard signed sliding window methods [4] since inversion is cheap in $G_{\Phi_k(p)}$. If $e \leq p$ then the best way to perform the exponentiation, using windows of width at least r, will take time

$$C_{\text{SSW}}(e) = \overline{\overline{S}}(1 + \log_2 e) + M\left(\frac{\log_2 e}{r+2} + (2^{r-2} - 1\right)$$

where $\overline{\overline{S}}$ denotes the time needed to perform a squaring operation in $G_{\Phi_k(p)}$. We also need to store 2^{r-2} elements during the exponentiation algorithm.

When $e \geq p$, as is the case in the final powering of the algorithm to compute the Tate pairing, one uses the fact that we can perform the Frobenius operation on $G_{\Phi_k(p)}$ for free. Thus we write e in base p, and perform a simultaneous exponentiation. Using the techniques of Avanzi [1], we can estimate the time needed to perform such a multi-exponentiation by

$$C_{\mathrm{bigSSW}}(e) = (d + \log_2 p)\overline{\overline{S}} + \left(d(2^{r-1} - 1) + \frac{\log_2 e}{r+2} - 1 \right) M$$

using windows of width r, where $d = \lceil \log_2 e / \log_2 p \rceil$. The precomputation storage can be reduced using techniques described in [2].

Note that for $k = 6$ one can also use XTR [11, 16] to gain a slight efficiency advantage over these methods if this is desirable [9], at a cost of altering particular protocols accordingly since multiplication is not straightforward in this case. For $k = 12$ and 24, one can also employ XTR defined over $\mathbb{F}_{p^2}$ and $\mathbb{F}_{p^4}$ respectively [12], however further work is required to determine if arithmetic can be made as efficient as in the original scheme for cases of interest in pairing-based cryptography.

3 Elliptic Curve Operations

In pairing based protocols we also need to conduct elliptic curve group operations. These are either on the main base curve $E(\mathbb{F}_p)$, or on the twisted curve $\overline{E}(\mathbb{F}_{p^{k/2}})$. We assume these curves take the form

$$E(\mathbb{F}_p) : Y^2 = X^3 - 3X + B$$

and

$$\overline{E}(\mathbb{F}_{p^{k/2}}) : \chi Y^2 = X^3 - 3X + B$$

where χ is a quadratic non-residue in $\mathbb{F}_{p^{k/2}}$ for which multiplication by χ is for free. Whether one should use affine or standard Jacobian projective coordinates are used, depends on the ratio ι/m and on the size of the $k/2$. It turns out that in some instances arithmetic in $\overline{E}(\mathbb{F}_{p^{k/2}})$ is better performed in affine coordinates. The various point addition and doubling times are summarized in the following table.

	$E(\mathbb{F}_p)$	$\overline{E}(\mathbb{F}_{p^{k/2}})$	
		Projective	Affine
Addition (A)	$12m + 4s$	$12\overline{M} + 4\overline{S}$	$2\overline{M} + 1\overline{S} + I_{k/2}$
Mixed Addition (A_M)	$8m + 3s$	$8\overline{M} + 3\overline{S}$	-
Doubling (D)	$4m + 4s$	$4\overline{M} + 4\overline{S}$	$2\overline{M} + 2\overline{S} + I_{k/2}$

We assume that exponentiation is performed via a signed sliding window method and mixed/affine addition

$$\mathrm{EC}_{\mathrm{SSW}}(e) = D(1 + \log_2 e) + A_M \left(\frac{\log_2 e}{r+2} + 2^{r-2} - 1 \right).$$

where the exact optimal choice for r depends on the size of e.

In some instances we wish to multiply by a random element in $\mathbb{Z}_n$, however in other instances (for example in the `MapToPoint` operation within the Boneh–Franklin encryption scheme [5]) we need to multiply by the cofactor. If we let $\log_2 p = \rho \cdot \log_2 n$ then the quantity 2^ρ measures how big the elliptic curve cofactor is for the curve $E(\mathbb{F}_p)$; a similar measure for the twisted curve is $(k\rho/2-1)\log_2 n$.

4 Application to Pairing Based Cryptography

In this section we wish to investigate the application of our techniques to pairing based cryptography in particular we focus on the case of non-supersingular curves of embedding degree $k \geq 6$. We follow the methodology of Koblitz and Menezes [10] which we recap on here, however we express our formulae in terms of total number of $\mathbb{F}_p$ operations as opposed to operations per bit. This is because this enables us to compare our sliding windows method in a more accurate manner and to also compare how other components of the protocols are affected by the choice of field.

Following Koblitz and Menezes we look at the cost of computing a Full-Miller operation or a Miller-Lite operation. The cost of these two operations, assuming projective coordinates are used, is

$$\begin{aligned} C_{\text{Full}} &= (km + 4\overline{S} + 6\overline{M} + S + M)\log_2 n \\ C_{\text{Lite}} &= (4s + (k+7)m + S + M)\log_2 n. \end{aligned}$$

In some instances one can more easily compute the Full-Miller algorithm by using affine coordinates in the main loop. In this case the cost is given by

$$C_{\text{Full}} = (2\overline{S} + 2\overline{M} + I_{k/2} + km + S + M)\log_2 n.$$

In computing the Tate pairing one executes one Miller-Lite operation and then an exponentiation for an exponent given by $\Phi_t(p)/n$ in the subgroup $G_{\Phi_k(p)}$. The bit length of $\Phi_t(p)/n$ is estimated by

$$\phi(k)\log_2 p - \log_2 n,$$

which can be expressed as

$$(\phi(k)\rho - 1)\log_2 n.$$

Thus a Tate pairing computation requires time

$$C_{\text{Tate}} = C_{\text{Luc}}(\Phi_t(p)/n) + C_{\text{Lite}}$$

or

$$C_{\text{Tate}} = C_{\text{bigSSW}}(\Phi_t(p)/n) + C_{\text{Lite}}.$$

In both of the above formulae for the Tate pairing we have ignored the inversion needed to take the input of Miller-Lite into the subgroup $G_{\Phi_k(p)}$, this is consistent with the analysis of Koblitz and Menezes but does slightly underestimate the cost in both cases.

The Weil pairing as pointed out by Koblitz and Menezes, could be more efficient, as it does not require an exponentiation by a large number. It requires time

$$C_{\text{Weil}} = C_{\text{Lite}} + C_{\text{Full}} + S.$$

We shall show in all cases of cryptographic relevance that the Weil pairing is always slower than the Tate pairing.

5 Results

In what follows we make the simplifying assumption that $m \approx s$. We wish to investigate what happens to pairing based protocols as the security level increases. We fix on the following parameter sizes to demonstrate the application of our model

Case	Security	k	$\log_2 n$	$\log_2 p$
A	80	6	160	160
B	128	6	256	512
C	128	12	256	256
D	192	6	384	1365
E	192	12	384	683
F	256	6	512	2560
G	256	12	512	1280
H	256	24	512	640

We do not discuss how such curves are generated, nor do we make use of special properties of the curves. For example when $k = 12$ with current technology one can only achieve $n \approx p$ by using the method of Barreto and Naehrig [3]. This results in curves with complex multiplication by $D = -3$, our analysis takes no account of the special optimizations which can be applied to such curves.

For each case we first present the operation counts, in terms of multiplications in $\mathbb{F}_p$, for the operations which do not appear to depend on the exact finite field we choose to use, namely the elliptic curve operations. We denote by (r) the size of the windows which produces the smallest operation count, the column n corresponds to exponentiation by a random integer of size n, whilst c corresponds to multiplication by the relevant cofactor. We limit window sizes to at most 9 bits, as otherwise the required look up table is likely to be prohibitively expensive. So as to get some idea about the relative merits of projective vs affine coordinates we made the assumption that $\iota/m \approx 10$ and in the table if the best performance for a given parameter set was using affine coordinates with give the multiplication count for this curve and denote this by an (A). We see that when $k \geq 12$ that it may make sense to use affine coordinates for the arithmetic in $\mathbb{G}_2$.

Case	$E(\mathbb{F}_p)$ n		$E(\mathbb{F}_p)$ c		$\overline{E}(\mathbb{F}_{p^{k/2}})$ n		$\overline{E}(\mathbb{F}_{p^{k/2}})$ c	
A	1614	(4)		-	8071	(4)	15739	(5)
B	2535	(5)	2535	(5)	12676	(5)	60767	(8)
C	2535	(5)		-	34813	(5)(A)	169000	(7)(A)
D	3760	(6)	9369	(6)	18802	(5)	172356	(8)
E	3760	(5)	2946	(5)	51801	(5)(A)	483113	(8)(A)
F	4973	(6)	19236	(7)	24865	(6)	329585	(9)
G	4973	(6)	7373	(6)	68671	(6)(A)	926142	(9)(A)
H	4973	(6)	1229	(4)	164573	(6)(A)	$2.2 \cdot 10^6$	(9)(A)

We now turn to the operations which depend on the field representation, i.e. whether we use a pairing friendly or a cyclotomic field extension. There are two operations which are important, the pairing computation itself and exponentiation in $G_{\Phi_k(p)}$ by an element of $\mathbb{Z}_n$. The pairing computation can itself either be computed by the Weil or Tate pairings. The results, in terms of estimated multiplications in $\mathbb{F}_p$, are presented in the following table. The (r) in the Tate column denotes the window size in the final exponentiation step, if Lucas sequences are faster we denote this by (L) and the operation count is for the application of Lucas sequences. In all cases using the Weil pairing method which used affine coordinates in the Full-Miller operation loop was the most efficient.

Case	Pairing Friendly: Pairing Weil	Pairing Friendly: Pairing Tate	Pairing Friendly: Exp in $G_{\Phi_k(p)}$	Cyclotomic Field: Pairing Weil	Cyclotomic Field: Pairing Tate	Cyclotomic Field: Exp in $G_{\Phi_k(p)}$
A	19855	9120 (L)	1411 (4)	18250	8247 (3)	1411 (4)
B	31759	18738 (5)	2195 (5)	29194	15916 (5)	2195 (5)
C	83757	43703 (4)	3502 (5)	-	-	-
D	47631	34664 (6)	3237 (5)	43786	29643 (6)	3237 (5)
E	125613	81751 (5)	5093 (5)	-	-	-
F	63503	56677 (6)	4263 (6)	58378	46431 (6)	4263 (6)
G	167469	127831 (6)	6633 (6)	-	-	-
H	446087	331078 (5)	13743 (6)	-	-	-

We see that for all fields the Tate pairing is always more efficient than the Weil pairing, at least for the security sizes that are likely to be used in practice. This is more due to the use of the efficient exponentiation algorithm as compared to the efficient squaring algorithm for $G_{\Phi_k(p)}$. In addition Lucas sequences are only more efficient than the signed sliding window method for very small security parameters.

To compare the different values of k we need to estimate the relative difference in time needed to compute a multiplication in $\mathbb{F}_p$, for the different sizes of p. If we assume that each finite field multiplication is performed using a standard interleaved Montgomery multiplication then the total number of 32-bit by 32-bit multiplication instructions which are needed to be performed per $\mathbb{F}_p$ multiplication is given by

$$2 \cdot t \cdot (t+1),$$

where $t = \log_2 p/32$. This leads us to the following table, where we present the number of 32-bit by 32-bit multiplication instructions needed for the various operations.

	Curve Operations				Pairing Friendly		Cyclotomic Field	
	$E(\mathbb{F}_p)$		$\overline{E}(\mathbb{F}_{p^{k/2}})$			Exp in	Pairing	Exp in
Case	n	c	n	c	Pairing	$G_{\Phi_k(p)}$	Pairing	$G_{\Phi_k(p)}$
A	$9.7 \cdot 10^4$	-	$4.8 \cdot 10^5$	$9.4 \cdot 10^4$	$5.4 \cdot 10^5$	$8.5 \cdot 10^4$	$4.9 \cdot 10^5$	$8.4 \cdot 10^5$
B	$1.3 \cdot 10^6$	$1.3 \cdot 10^6$	$6.8 \cdot 10^6$	$3.3 \cdot 10^7$	$1.0 \cdot 10^7$	$1.1 \cdot 10^6$	$8.6 \cdot 10^6$	$1.2 \cdot 10^6$
C	$3.6 \cdot 10^5$	-	$5.0 \cdot 10^6$	$2.4 \cdot 10^7$	$6.2 \cdot 10^6$	$1.1 \cdot 10^6$	-	-
D	$1.4 \cdot 10^7$	$3.4 \cdot 10^7$	$7.0 \cdot 10^7$	$6.4 \cdot 10^8$	$1.3 \cdot 10^8$	$1.2 \cdot 10^7$	$1.1 \cdot 10^8$	$1.2 \cdot 10^7$
E	$3.6 \cdot 10^6$	$2.8 \cdot 10^6$	$4.9 \cdot 10^7$	$4.6 \cdot 10^8$	$7.8 \cdot 10^7$	$4.8 \cdot 10^6$	-	-
F	$6.4 \cdot 10^7$	$2.5 \cdot 10^8$	$3.2 \cdot 10^8$	$4.3 \cdot 10^9$	$7.3 \cdot 10^8$	$5.5 \cdot 10^7$	$6.0 \cdot 10^8$	$5.5 \cdot 10^7$
G	$1.6 \cdot 10^7$	$2.4 \cdot 10^7$	$2.2 \cdot 10^8$	$3.0 \cdot 10^9$	$4.2 \cdot 10^8$	$2.2 \cdot 10^7$	-	-
H	$4.0 \cdot 10^6$	$1.0 \cdot 10^6$	$1.4 \cdot 10^8$	$1.8 \cdot 10^9$	$2.8 \cdot 10^8$	$1.1 \cdot 10^7$	-	-

From the table one can see that the main advantage in using values of k which are larger than 6 is in the basic elliptic curve operations over $\mathbb{F}_p$, rather than in the pairing computation. For the pairing computation one gains some advantage for using large values of k, but this is not as pronounced as for the elliptic curve operations.

However, elliptic curve operations are relatively cheap in comparison to pairing calculation and so the performance improvement will not be so pronounced. Except, for protocols in which one party only needs to perform elliptic curve operations in $\mathbb{F}_p$, such as the encryptor in the Sakai–Kasahara KEM [7]. It does however imply that for pairing based protocols one should not neglect selecting parameter values which speed up the elliptic curve operations and not just the pairing calculation.

However, our estimates are on the conservative side for arithmetic in cyclotomic fields at high security levels. This is for a number of reasons. The overhead in not having to deal with different values of k and f_0 means that the library overhead in using cyclotomic fields of degree six will be less than for pairing friendly fields. Recall, we have not given accurate cycle counts, but simply estimated the number of multiplication instructions needed.

One should also bear in mind that larger values of k mean that one can shrink the bandwidth required in communication if one is communicating elements in $E(\mathbb{F}_p)$, since a larger value of k corresponds to a smaller value of p.

6 Acknowledgements

The authors would like to thank Mike Scott, for a some insightful comments which improved this paper.

References

1. R.M. Avanzi. *On Multi-exponentiation in Cryptography.* In Cryptology ePrint Archive, Report 2002/154, 2002.

2. R. M. Avanzi and P. Mihailescu. *Generic efficient arithmetic algorithms for PAFFs (Processor Adequate Finite Fields) and related algebraic structures.* In Selected Areas in Cryptology – SAC 2003, Springer-Verlag LNCS 3006, 320–334, 2004.

3. P.S.L.M. Barreto and M. Naehrig. Pairing-friendly elliptic curves of prime order. Preprint, 2005.

4. I.F. Blake, G. Seroussi and N.P. Smart. *Elliptic Curves in Cryptography.* Cambridge University Press, 1999.

5. D. Boneh and M. Franklin. *Identity-based encryption from the Weil pairing.* SIAM Journal of Computing, **32**, 586–615, 2003.

6. F. Brezing and A. Weng. *Elliptic Curves Suitable for Pairing Based Cryptography.* Designs, Codes and Cryptography, **37**, 133–141, 2005.

7. M. Cheng, L. Chen, J. Malone-Lee and N.P. Smart. *An Efficient ID-KEM Based On The Sakai-Kasahara Key Construction.* To appear, 2006.

8. M. van Dijk, R. Granger, D. Page, K. Rubin, A. Silverberg, M. Stam and D. Woodruff. *Practical cryptography in high dimensional tori.* In Advances in Cryptology – EUROCRYPT 2005, Springer-Verlag LNCS 3494, 234–250, 2005.

9. R. Granger, D. Page and M. Stam. *A Comparison of CEILIDH and XTR.* In Algorithmic Number Theory Symposium – ANTS VI, Springer-Verlag LNCS 3076, 235–249, 2004.

10. N. Koblitz and A. Menezes. *Pairing-based Cryptography at High Security Levels.* In Cryptography and Coding, Springer-Verlag LNCS 3796, 13–36, 2005.

11. A. K. Lenstra and E. Verheul. *The XTR Public Key System.* In Advances in Cryptology – CRYPTO 2000, Springer LNCS 1880, 1–19, 2000.

12. S. Lim, S. Kim, I. Yie, J. Kim and H. Lee. *XTR extended to* $GF(p^{6m})$. In Selected Areas in Cryptography – SAC 2001, Springer LNCS 2259, 301–312, 2001.

13. M. Scott and P.S.L.M. Barreto. *Compressed Pairings.* In Advances in Cryptology – CRYPTO 2004, Springer-Verlag LNCS 3152, 140–156, 2004.

14. M. Stam. *Speeding up Subgroup Cryptosystems.* PhD Thesis, T.U. Eindhoven, 2003.

15. M. Stam and A. Lenstra. *Efficient Subgroup Exponentiation in Quadratic and Sixth Degree Extensions.* In Cryptographic Hardware and Embedded Systems – CHES 2002, Springer-Verlag LNCS 2523, 318–332, 2002.

16. M. Stam and A. K. Lenstra. *Speeding Up XTR.* In Advances in Cryptology – ASIACRYPT 2001, Springer LNCS 2248, 125–143, 2001.

A Squaring formulae for $G_{\Phi_k(p)}$ for pairing friendly fields and $k = 12$ and $k = 24$

A.1 $k = 12$

$$
\begin{aligned}
b_0 &= 3\,{a_0}^2 - 6\,f_0 a_3 a_9 - 3\,f_0 {a_6}^2 - 2\,a_0,\\
b_1 &= -6\,f_0 a_{11} a_2 - 6\,f_0 a_5 a_8 + 2\,a_1,\\
b_2 &= 3\,{a_1}^2 - 6\,f_0 a_4 a_{10} - 3\,f_0 {a_7}^2 - 2\,a_2,\\
b_3 &= -6\,f_0 a_6 a_9 + 6\,a_0 a_3 + 2\,a_3,\\
b_4 &= -3\,f_0 {a_8}^2 - 6\,f_0 a_{11} a_5 + 3\,{a_2}^2 - 2\,a_4,\\
b_5 &= -6\,f_0 a_7 a_{10} + 6\,a_4 a_1 + 2\,a_5,\\
b_6 &= -3\,f_0 {a_9}^2 + 6\,a_0 a_6 + 3\,{a_3}^2 - 2\,a_6,\\
b_7 &= 6\,a_5 a_2 - 6\,f_0 a_{11} a_8 + 2\,a_7,\\
b_8 &= -3\,f_0 {a_{10}}^2 + 3\,{a_4}^2 + 6\,a_7 a_1 - 2\,a_8,\\
b_9 &= 6\,a_6 a_3 + 6\,a_0 a_9 + 2\,a_9,\\
b_{10} &= -3\,f_0 {a_{11}}^2 + 3\,{a_5}^2 + 6\,a_8 a_2 - 2\,a_{10},\\
b_{11} &= 6\,a_{10} a_1 + 6\,a_7 a_4 + 2\,a_{11}.
\end{aligned}
$$

A.2 $k = 24$

$$
\begin{aligned}
b_0 &= -2\,a_0 + 3\,a_0{}^2 - 3\,f_0 a_{12}{}^2 - 6\,f_0 a_3 a_{21} - 6\,f_0 a_6 a_{18} - 6\,f_0 a_9 a_{15},\\
b_1 &= 2\,a_1 - 6\,f_0 a_2 a_{23} - 6\,f_0 a_5 a_{20} - 6\,f_0 a_8 a_{17} - 6\,f_0 a_{11} a_{14},\\
b_2 &= -2\,a_2 - 6\,f_0 a_4 a_{22} - 6\,f_0 a_{10} a_{16} - 3\,f_0 a_{13}{}^2 - 6\,f_0 a_7 a_{19} + 3\,a_1{}^2,\\
b_3 &= 2\,a_3 - 6\,f_0 a_9 a_{18} + 6\,a_0 a_3 - 6\,f_0 a_6 a_{21} - 6\,f_0 a_{12} a_{15},\\
b_4 &= -2\,a_4 - 3\,f_0 a_{14}{}^2 - 6\,f_0 a_5 a_{23} - 6\,f_0 a_8 a_{20} - 6\,f_0 a_{11} a_{17} + 3\,a_2{}^2,\\
b_5 &= 2\,a_5 - 6\,f_0 a_{13} a_{16} - 6\,f_0 a_{10} a_{19} + 6\,a_4 a_1 - 6\,f_0 a_7 a_{22},\\
b_6 &= -2\,a_6 + 6\,a_0 a_6 - 3\,f_0 a_{15}{}^2 - 6\,f_0 a_{12} a_{18} - 6\,f_0 a_9 a_{21} + 3\,a_3{}^2,\\
b_7 &= 2\,a_7 - 6\,f_0 a_{14} a_{17} - 6\,f_0 a_{11} a_{20} - 6\,f_0 a_8 a_{23} + 6\,a_2 a_5,\\
b_8 &= -2\,a_8 - 6\,f_0 a_{10} a_{22} - 6\,f_0 a_{13} a_{19} + 3\,a_4{}^2 + 6\,a_7 a_1 - 3\,f_0 a_{16}{}^2,\\
b_9 &= 2\,a_9 + 6\,a_0 a_9 - 6\,f_0 a_{12} a_{21} - 6\,f_0 a_{15} a_{18} + 6\,a_6 a_3,\\
b_{10} &= -2\,a_{10} - 3\,f_0 a_{17}{}^2 - 6\,f_0 a_{11} a_{23} - 6\,f_0 a_{14} a_{20} + 6\,a_2 a_8 + 3\,a_5{}^2,\\
b_{11} &= 2\,a_{11} - 6\,f_0 a_{13} a_{22} - 6\,f_0 a_{16} a_{19} + 6\,a_{10} a_1 + 6\,a_7 a_4,\\
b_{12} &= -2\,a_{12} + 6\,a_0 a_{12} + 6\,a_9 a_3 - 3\,f_0 a_{18}{}^2 - 6\,f_0 a_{15} a_{21} + 3\,a_6{}^2,\\
b_{13} &= 2\,a_{13} - 6\,f_0 a_{17} a_{20} - 6\,f_0 a_{14} a_{23} + 6\,a_5 a_8 + 6\,a_2 a_{11},\\
b_{14} &= -2\,a_{14} + 6\,a_{13} a_1 + 6\,a_{10} a_4 - 3\,f_0 a_{19}{}^2 - 6\,f_0 a_{16} a_{22} + 3\,a_7{}^2,\\
b_{15} &= 2\,a_{15} + 6\,a_0 a_{15} + 6\,a_{12} a_3 + 6\,a_9 a_6 - 6\,f_0 a_{18} a_{21},\\
b_{16} &= -2\,a_{16} - 3\,f_0 a_{20}{}^2 - 6\,f_0 a_{17} a_{23} + 6\,a_5 a_{11} + 6\,a_2 a_{14} + 3\,a_8{}^2,\\
b_{17} &= 2\,a_{17} - 6\,f_0 a_{19} a_{22} + 6\,a_{13} a_4 + 6\,a_{10} a_7 + 6\,a_{16} a_1,\\
b_{18} &= -2\,a_{18} + 6\,a_0 a_{18} + 6\,a_{15} a_3 + 6\,a_{12} a_6 - 3\,f_0 a_{21}{}^2 + 3\,a_9{}^2,\\
b_{19} &= 2\,a_{19} - 6\,f_0 a_{20} a_{23} + 6\,a_5 a_{14} + 6\,a_8 a_{11} + 6\,a_2 a_{17},\\
b_{20} &= -2\,a_{20} - 3\,f_0 a_{22}{}^2 + 6\,a_{19} a_1 + 6\,a_{16} a_4 + 6\,a_{13} a_7 + 3\,a_{10}{}^2,\\
b_{21} &= 2\,a_{21} + 6\,a_0 a_{21} + 6\,a_{18} a_3 + 6\,a_{15} a_6 + 6\,a_{12} a_9,\\
b_{22} &= -2\,a_{22} - 3\,f_0 a_{23}{}^2 + 6\,a_5 a_{17} + 6\,a_8 a_{14} + 6\,a_2 a_{20} + 3\,a_{11}{}^2,\\
b_{23} &= 6\,a_{16} a_7 + 6\,a_{13} a_{10} + 6\,a_{19} a_4 + 6\,a_{22} a_1 + 2\,a_{23}.
\end{aligned}
$$