Pairing Computation for Elliptic Curves with Embedding Degree 15

Nadia El Mrabet, Sorina Ionica

Pairing computation for elliptic curves with embedding degree 15

Nadia EL MRABET[1] & Sorina IONICA[2]
1 LIRMM-I3M-CNRS, Universit Montpellier 2
2 PRISM, Universit de Versailles

No Institute Given

Abstract. This paper presents the first study of pairing computation on curves with embedding degree 15. We show that pairing computation on these curves has loop length $r^{1/8}$ and we use a twist of degree 3 to perform most of the operations in $\mathbb{F}_p$ or $\mathbb{F}_{p^5}$. Furthermore, we present an original arithmetic for extension fields of degree 5.
Key-words: Pairing based cryptography, Pairing computation, Arithmetic, Interpolation, Elliptic Curves, Embedding degree.

1 Introduction

Pairings on elliptic curves were introduced by André Weil in 1948 in mathematics [17], but their utilization in cryptography is actually quite recent. They were first used for cryptanalytic purposes, i.e. attacking the discrete logarithm problem on the elliptic curve [1], but nowadays they are also used as bricks for building new cryptographic protocols such as the tripartite Diffie-Hellman protocol [10], identity-based encryption [3], short signatures [4], and others..

A pairing is a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_3$, where $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_3$ are groups of large prime order r. Known pairings on elliptic curves, i.e. the Weil, Tate, Eta and Ate pairings map to the multiplicative group of the minimal extension of the ground field $\mathbb{F}_p$ which contains the r^{th} roots of unity. The degree of this extension is called the embedding degree with respect to r. The most efficient known method used for pairing computation is Miller's algorithm, whose performance heavily relies on efficient arithmetic of this extension field. It follows that for efficient pairing computation we need curves with embedding degree rather small.

On the other hand, latest research in efficient computation of pairings focused on the reduction of the loop length in Miller's algorithm. It was proved in [16] that on most existing constructions of ordinary curve families, the complexity of Miller's algorithm is $O(r^{1/\varphi(k)})$, where k is the embedding degree and φ the Euler function. Consequently, for a fixed level of security and therefore a fixed bit length of r, pairing computation might turn out to be faster on curves with embedding degrees such that the integer $\varphi(k)$ is large.

Moreover, in practice we are looking for curves for which the following value $\rho = \frac{\log p}{\log r}$ is as small as possible, in order to save bandwidth during the calculation.

According to [7] curves of embedding degree $k = 15$ achieve a security level of 256 bits for curves with $\rho \approx 2$, and a security level of 112 and 128 bits for $\rho \approx 1$. Consequently, for the future security level, pairing computations on curves with embedding degree 15 must be taken into consideration.

In this paper, we give the first efficient pairing computation for curves of embedding degree 15. We show that existing constructions of families of curves of degree 15 and j-invariant 0 present multiple advantages. First of all, we show that pairing computation on these curves has loop length $r^{1/8}$, which is an important gain when compared to $r^{1/4}$, which is the loop length of Miller's algorithm for curves of embedding degree 12. Secondly, we show that by using twists of degree 3 we manage to perform most of the operations in $\mathbb{F}_p$ or $\mathbb{F}_{p^5}$. By making use of an interpolation technique, we also improve the arithmetic of $\mathbb{F}_{p^5}$ in order to get better results. Moreover, denominator computation and the final inversion can be

avoided by making use of the twist. Our results show that by choosing the optimal arithmetic on $\mathbb{F}_{p^5}$ and $\mathbb{F}_{p^{15}}$, curves of embedding degree 15 are competitive when compared to Barreto-Naehrig curves of embedding degree 12 and should thus be considered for cryptographic use.

The remainder of this paper is organized as follows: Section 2 gives the definition and important properties of pairings. In Section 3 we establish the optimal computation of the pairing on curves with embedding degree 15. In Section 4 we describe an efficient multiplication based on the interpolation for the field $\mathbb{F}_{p^5}$. Finally, we conclude in Section 5 by giving a global evaluation of the number of operations needed to compute the pairing and by comparing our results to performances obtained on Barreto-Naehrig curves, which are considered as standard at the time this paper was written.

2 Background on pairings

In this section we give a brief overview of the definitions of pairings on elliptic curves and of Miller's algorithm [14] used in pairing computation. Let E be an elliptic curve defined by the Weierstrass equation $y^2 = x^3 + ax + b$ and r a prime factor of $\#(E)$. Suppose r^2 does not divide $\#(E)$ and k be the *embedding degree* with respect to r, i.e. the smallest integer such that r divides $p^k - 1$.

Definition 1. *A pairing is a bilinear and non degenerate function:*

$$\begin{aligned} e : G_1 \times G_2 &\to G_3 \\ (P, Q) &\to e(P, Q) \end{aligned}$$

where G_1 and G_2 are subgroups of order r on the elliptic curve and $\mathbb{G}_3$ is generally μ_r, the subgroup of the r^{th} roots of the unity in $\mathbb{F}_{p^k}$. In general we take $G_1 = E(\mathbb{F}_p)[r]$ and $G_2 \subset E(\mathbb{F}_{p^k})[r]$, where we denote by $E(K)[r]$ the subgroup of points of order r of the elliptic curve E over the field K. We also denote $E[r]$ the subgroup of points of order r defined over the algebraic closure of $\mathbb{F}_p$.

Let $P \in G_1$, $Q \in G_2$. The goal of Miller's algorithm is to first construct a rational function $f_{s,P}$ associated to the point P and to some integer s and to secondly evaluate this function at point Q (in fact at a divisor associated to this point). The function $f_{s,P}$ is such that the divisor associated to it is:

$$div(f_{s,P}) = s(P) - (sP) - (s-1)(P_\infty).$$

Suppose we want to compute the sum of iP and jP. Take h_1 the line going through iP and jP and h_2 the vertical line through $(i+j)P$. Miller's idea was to make use of the following relation

$$f_{i+j,P} = f_{i,P} f_{j,P} \frac{h_1}{h_2}, \tag{1}$$

in order to compute $f_{s,P}$ iteratively. Moreover, Miller's algorithm uses the double-and-add method to compute $f_{s,P}$ in $\log_2(s)$ operations.

The Tate pairing , denoted e_{Tate}, is defined by:

$$\begin{aligned} G_1 \times G_2 &\mapsto G_3 \\ (P, Q) &\mapsto e_{Tate}(P, Q) = f_{r,P}(Q). \end{aligned}$$

Here, the function $f_{r,P}$ is normalized, i.e. $(u_0^r f_{r,P})(P_\infty) = 1$ for u_0^r some $\mathbb{F}_p$-rational uniformizer at P_∞. This pairing is only defined up to a representative of $(\mathbb{F}_{p^k})^r$. In order to obtain a unique value we raise it to the power $\frac{p^k-1}{r}$, obtaining an r-th root of unity that we call the reduced Tate pairing

$$\hat{e}_{Tate}(P, Q) = f_{r,P}(Q)^{\frac{p^k-1}{r}}.$$

Twisted Ate pairing We begin with the following definition.

Definition 2. *Let E, E' be elliptic curves over F_p. Then E' is called a twist of degree d if there exists an isomorphism $\phi_d : E' \to E$ defined over $\mathbb{F}_{p^d}$ and d is minimal.*

Suppose now that E admits a twist E' defined over $\mathbb{F}_{p^{k/d}}$ of degree d, with $d|k$. We set $m = \gcd(k, d)$ and $e = k/m$. We denote by π_p be the Frobenius map over E: $\pi_p : E \to E : (x, y) \to (x^p, y^p)$. We consider $G_1 := E[r] \cap \mathrm{Ker}(\pi_p - [1])$ and $G_2 := E[r] \cap \mathrm{Ker}(\pi_p - [p])$. Let t be the trace of the Frobenius map and $T = t - 1$. Then for $P \in G_1, Q \in G_2$ we get the following relation [9]:

$$e_{Tate}(P, Q)^L = f_{T^e,P}(Q)^{c(p^k-1)/N},$$

where $N = \gcd(T^k - 1, p^k - 1), T^k - 1 = LN, c = \sum_{i=0}^{m-1} T^{e(m-1-i)} p^{ei} \equiv mp^{e(m-1)} \bmod r$.

So for $T^k - 1 \neq 0$, we have $r \nmid L$ and we can define a non-degenerate, bilinear pairing that we call the *reduced twisted Ate pairing* as $f_{T^e,P}(Q)^{(p^k-1)/r}$. For curves with small trace of the Frobenius, it is clear that this pairing should be preferred to the Tate pairing, as the loop in Miller's algorithm will be shorter.

Other variants of twisted Ate pairing were obtained in [13] replacing T^e with T^{ie} respectively, for any $i \in \mathbb{Z}$. All these variants were given in order to find the smallest possible λ determining the length of the loop in Miller's algorithm.

Optimal pairing Vercauteren exploits all these ideas in [16] and introduces the concept of optimal pairing. The basic idea is to look for a multiple $\lambda = mr$ such that $\lambda = \sum_{i=0}^{l} c_i T^{ie}$ with small coefficients. The following result is a slightly modified variant of a theorem from [16].

Theorem 1. *Let $\lambda = mr$ with $r \nmid m$ and write $\lambda = \sum_{i=0}^{l} c_i p^{ie}$ then*

$$\begin{aligned} a_{[c_0,\ldots,c_l]} : G_1 \times G_2 &\to \mu_r \\ (P, Q) &\to \left(\prod_{i=0}^{l} f_{c_i,P}^{p^{ie}}(Q) \prod_{i=0}^{l-1} \frac{l_{[s_{i+1}]P,[c_i p^{ie}]P}(Q)}{v_{[s_i]P}(Q)} \right) \end{aligned} \tag{2}$$

with $s_i = \sum_{j=i}^{l} c_j p^{je}$, defines a bilinear pairing. Furthermore, if

$$mkp^{e(k-1)} \neq (p^k - 1)/r \cdot \sum_{i=0}^{l} i c_i p^{e(i-1)} \bmod r,$$

then the pairing is non-degenerate.

The proof of this theorem is similar to the one of Theorem 1 in [16].

Security aspect The security of a pairing based cryptosystem relies on two parameters: the bit length of r, $\log_2 r$ and the bit size of the extension field $k\log_2 p$. These parameters have to be chosen high enough to ensure that the discrete logarithm problem will be hard in both the subgroup of points of order r of the curve and the finite field $\mathbb{F}_{p^k}$. Considering the complexity of index calculus attacks ($O(e^{\sqrt{log(p)log(log(p))}})$) against the discrete logarithm problem in finite fields ($O(\sqrt{r})$), while the security level will increase, the bound on $k\log_2 p$ is expected to grow faster than the bound on $\log_2 r$. On the other hand, in practice we are looking for curves for which the following value

$$\rho = \frac{\log p}{\log r}$$

is as small as possible, in order to save bandwith during the calculation. This is due to the fact that for a fixed level of security (i. e. fixed size of r), efficient implementation of the pairing depends on the size of the ground field, i.e. on the size of p. So taking greater k is a better solution than increasing the bit length of p.

3 Optimal pairing for $k = 15$

A first method that could be used in order to build curves with $k = 15$ is the Cocks-Pinch method [5]. This method generates curves with arbitrary r and $\rho \approx 2$. Duan and all. [6] showed that by using the Brezing-Weng method we can actually do better. They generated a family of curves with j-invariant 0 and embedding degree 15 and $\rho \approx 1,5$. This family is given by the following polynomials:

$$
\begin{aligned}
p &= 1/3 * x^{12} - 2/3 * x^{11} + 1/3 * x^{10} + 1/3 * x^7 - 2/3 * x^6 + 1/3 * x^5 + 1/3 * x^2 + 1/3 * x + 1/3 \\
r &= x^8 - x^7 + x^5 - x^4 + x^3 - x + 1 \\
t &= x + 1.
\end{aligned}
$$

The remainder of this paper will present efficient computation of pairings on this family of curves. To emphasize the performance of our suggestion, we compare our results to those resulting from efficient implementation of pairings on Barreto-Naehrig curves [2]. We briefly remind that these are curves of embedding degree 12 and j-invariant 0, given by the following parametrization:

$$
\begin{aligned}
p &= 36x^4 + 36x^3 + 24x^2 + 6x + 1 \\
r &= 36x^4 + 36x^3 + 18x^2 + 6x + 1 \\
t &= 6x^2 + 1.
\end{aligned}
$$

These curves are preferred in cryptographic applications because they have the $\rho \sim 1$ and most operations during the pairing computation are done in $\mathbb{F}_p$ or $\mathbb{F}_{p^2}$, thanks to the existence of a twist of degree 6. The complexity of Miller iteration as taken from [8] is $5S_p + 18M_p + S_{p^{12}} + M_{p^{12}}$.

Twists of degree 3 Let E be an elliptic curve of j-invariant 0, defined over $\mathbb{F}_p$. Suppose its equation is $y^2 = x^3 + b$, with $b \in \mathbb{F}_p$. Consider E over the extension field $\mathbb{F}_{p^{k/3}}$. Then it admits a cubic twist E' of equation $y^2 = x^3 + \frac{b}{D}$, with D not a cubic residue $D \in \mathbb{F}_{q^{k/3}}$. The morphism

$$\Phi_3 : E' \to E : \Phi_3(x, y) = (xD^{1/3}, yD^{1/2})$$

maps points in $E'(\mathbb{F}_{p^{k/3}})$ to points in $E(\mathbb{F}_{p^k})$. In particular, as $r|\#E'$, we may choose Q, the generator of G_2, as the image of an r order point under this morphism : $Q = \Phi_3(Q')$, with $Q' \in E'(\mathbb{F}_{p^{k/3}})$. So $Q = (D^{1/3}x, D^{1/2}y)$, with $x, y \in \mathbb{F}_{p^{k/3}}$. As we will see later, for $k = 15$ this will imply that most operations in pairing computation on $G_1 \times G_2$ (or $G_2 \times G_1$) are to be done in $\mathbb{F}_p$ or $\mathbb{F}_{p^5}$.

Optimal pairing for $k = 15$ We can easily see that for the family of curves with $k = 15$ described above, the length of the Miller's loop is $\frac{3}{8}\log_2 r$. We will show that the optimal pairing for this family has Miller's loop length $\frac{\log_2 r}{8}$. Indeed, we look for a combination $mr = \sum_{i=0}^{l} c_i T^{ie}$. Since $T \equiv p \bmod r$ and the powers of p^e are related modulo r via $\Phi_k(p^e) \equiv 0 \bmod r$, it follows that we can consider only the powers T^{ie}, for $i = 0, ..., \phi(k) - 1$ in Theorem 1. A similar result can be obtained on $G_1 \times G_2$ by considering combinations of the twisted Ate pairing and its variants $\lambda = \sum_{i=0}^{l} c_i T^{ei}$, for $i = 0, ..., \phi(k)-1$. Small coefficients c_i will be obtained by looking for short vectors in the following lattice for the Twisted Ate:

$$
\begin{pmatrix}
r & 0 & 0 & ... & 0 \\
-T^e & 1 & 0 & ... & 0 \\
. & . & . & & . \\
-T^{e(\phi(k)-1)} & 0 & ... & 0 & 1
\end{pmatrix}
$$

This matrix is of volume r, so by Minkowski ([15]) there exists a short vector V in the lattice with $||V||_\infty \leq r^{1/\phi(15)}$, where $||V||_\infty = \max_i |v_i|$. So the shortest vector in this lattice gives the optimal pairing, defined by Equation 2. Our computations produced the following short vector with only one coefficient of size x:

$$V(x) = [2187, -2187, 0, 81, -27, 9, 0, x-1].$$

While the Tate and twisted Ate pairings have loop length $\log r$ for Barreto-Naehrig curves, a search for the optimal vector on these curves gives, for example, $[1, x-1, -(x-1), x]$. So the loop length of Miller's algorithm is $\frac{\log_2 r}{4}$.

Denominator elimination in pairing computation We use an idea given in [12]. We observe that the expression of line h_2 in Equation (1) can be written as:

$$x_T - x_Q = \frac{x_T^3 - x_Q^3}{x_T^2 + x_T x_Q + x_Q^2} = \frac{y_P^2 - y_Q^2}{x_T^2 + x_T x_Q + x_Q^2}.$$

The element $(y_Q^2 - y_T^2)$ is in $\mathbb{F}_{p^5}$ and can be forgotten during the computation of the pairing, because of the final exponentiation. Indeed, $p^5 - 1$ is a divisor of $\frac{p^{15}-1}{r}$ so multiplication by this term can be omitted. So at each iteration in Miller's algorithm loop it suffices to multiply by $x_P^2 + x_P x_Q + x_Q^2$, instead of dividing by h_2. This saves operations, as we no longer need to compute denominators at each step and also avoids the final inversion, which is important on restricted devices.

Operation count Suppose we want to compute $f_{T^e,P}(Q)$, with $P \in G_1$ and $Q = (x_Q, y_Q) \in G_2$ Then one of the most efficient known ways of computing the pairing is to use Jacobian coordinates, as stated in [11] and [8]. The point $iP = (X, Y, Z)$ in Jacobian coordinates represents the affine point $(X/Z^2, Y/Z^3)$ on the elliptic curve. Due to the denominator elimination, the doubling step of the Miller loop using Equation (1) becomes:

$$\begin{aligned} (2i)P &\leftarrow 2 \cdot (iP) \\ f_{2i,P} &\leftarrow f_{i,P}^2 h_1(Q) S_T(Q) \end{aligned}$$

where $h_1 = Z_3 Z^2 y_Q - 3X^2(Z^2 x_Q - X) - 2Y^2$ and $S_T(Q) = Z^4 x_Q^2 + X Z^2 x_Q + X^2$ and We compute $(2i)P = (X_3, Y_3, Z_3)$ as

$$\begin{aligned} X_3 &= 9X^4 - 8XY^2, \\ Y_3 &= 3X^2(4X_1 Y_1^2 - X_3) - 8Y^4, \\ Z_3 &= 2YZ. \end{aligned}$$

We perform the operations in the following order:

$$\begin{aligned} A &= Y^2 \\ B &= 4X \cdot A \\ C &= X^2 \\ X_3 &= -2B + 3C^2 \\ Y_3 &= -8A^2 + 3C(B - X_3) \\ D &= Z^2 \\ Z_3 &= (Y+Z)^2 - A - D \end{aligned}$$

$$\begin{aligned} F &= Z_3 \cdot D \\ E &= CD \\ h_1 &= 2F \cdot y_Q - A - 3E x_Q - 3b \\ F &= D^2 \\ G &= (X + Z^2)^2 - X^2 - F \\ S_T &= C + X x_Q + F x_Q^2 \end{aligned}$$

The square x_Q^2 can be precomputed, as coordinates of point Q do not change during the pairing computations. A count of the operations for the entire doubling step gives $8S_p + 24M_p + S_{p^{15}} + 2M_{p^{15}}$, where S_p (resp. M_p) represents a square (resp. multiplication) in $\mathbb{F}_p$ and $S_{p^{15}}$ (resp. $M_{p^{15}}$) represents a square (resp. multiplication) in $\mathbb{F}_{p^{15}}$.

3.1 First comparison

We compare the complexity of computing the Twisted Ate pairing for embedding degree 12 and 15, using the multiplication of Karatsuba and Toom Cook in the extension fields. The cost of a multiplication in an extension field are given in Table 1. We denote M_{p^e} a multiplication in the extension field $\mathbb{F}_{p^e}$ and we give its complexity in number of multiplications (M_p) and additions (A_p) in $\mathbb{F}_p$ using the Karatsuba and Toom Cook method. The resulting comparison for the Miller loop are given in Table 2. We give just the number of multiplications in $\mathbb{F}_p$ needed for a Miller loop.

Table 1. A performance evaluation: arithmetic of $\mathbb{F}_{p^{15}}$ versus arithmetic of $\mathbb{F}_{p^{12}}$

M_{p^2}	M_{p^3}	M_{p^5}	$M_{p^{12}}$	$M_{p^{15}}$
$3M_p + 4A_p$	$5M_p + 20A_p$	$13M_p + 58A_p$	$45M_p + 180A_p$	$65M_p + 390A_p$

Table 2. A performance evaluation: curves with embedding degree 15 versus Barreto-Naehrig curves

AES security	bit length of r	twisted Ate	
		k=15	k=12
80	160	4 175	4 020
128	256	6 680	6 432
192	384	10 020	9 648
256	512	13 360	12 864

Using the multiplication of Karatsuba and Toom Cook, pairing computation on curves with embedding degree 15 is less efficient than computation of pairing over Barreto-Naehrig curves. The pairing computation for $k = 15$ is 4% more expensive than computation for $k = 12$. This difference comes from the arithmetic in intermediate fields. For curves of embedding degree 12, major computations are done in $\mathbb{F}_{p^2}$, while for curves with $k = 15$ the computations are essentially done in $\mathbb{F}_{p^5}$. As Karatsuba is optimal for extension of degree 2, it seems quite natural to obtain this result. We propose in the next Section an improvement of the arithmetic on $\mathbb{F}_{p^5}$ using the Newton interpolation method to compute a multiplication between two elements of $\mathbb{F}_{p^5}$.

4 Finite field arithmetic

In cryptography, and more generally in arithmetic, we need an efficient polynomial multiplication. The optimization can be in time or elementary operations. Pairing Based Cryptography (PBC) follows the same rules as PBC involves polynomial computations. Indeed A and $B \in \mathbb{F}_{p^k}$ are represented as polynomial of degree $(k-1)$ in γ, with γ a root in $\mathbb{F}_{p^k}$ of a polynomial of degree k irreducible in $\mathbb{F}_p$. So far as the irreducible polynomial is $X^k - \beta$, with $\beta \in \mathbb{F}_p$. Considering $k = 5$, this condition is true for every prime p such that $p \equiv 1 \bmod(5)$.

In extension of degree 2 or 3 the Karatsuba and Toom Cook multiplications are the more efficient. For higher degree extension, one can use tower field extensions [11] and apply Karatsuba and Toom Cook [18],

or multiplication by interpolation [18]. Generally, interpolation method have the drawback to increase the number of additions during a multiplication. We present a multiplication by Newton interpolation such that the additional additions improve the total complexity of the multiplication in $\mathbb{F}_{p^5}$ compared to the Karatsuba multiplication.

4.1 Interpolation

We denote $A(X) = a_0 + a_1X + \ldots + a_{k-1}X^{k-1}$, $B(X) = b_0 + b_1X + \ldots + b_{k-1}X^{k-1}$, the polynomial obtained by substituting γ by X in the expressions of A and B in $\mathbb{F}_{p^k}$. Multiplications by interpolation follow this step:

- Find $2k-1$ different values in $\mathbb{F}_p$
  - $\alpha_0, \alpha_1, \ldots, \alpha_{2k-2}$.
- Evaluate the polynomials $A(X)$ and $B(X)$ at this $2k-1$ values.
  - Stock $A(\alpha_0), \ldots, A(\alpha_{2k-2}), B(\alpha_0), \ldots, B(\alpha_0)$.
- Compute $C(X) = A(X) \times B(X)$ at this $2k-1$ values
  - $C(\alpha_i) = A(\alpha_i)B(\alpha_i)$.
- Interpolate $C(X)$ polynomial of degree $2k-2$
  - either with Lagrange or Newton interpolation.

We describe out method of multiplication using the Newton interpolation, which is more efficient for our purpose than Lagrange interpolation [18]. The use of FFT [18] is not interesting in our case. Indeed, during a FFT multiplication, we have to multiply by roots of unity, as we do not have any control on the characteristic p we work with, the roots of unity do not necessarily have a sparse representation, even after recoding, and then multiplications by this roots are expensive. Furthermore, the choice of value of interpolation in Section 4.3 is not interesting for a FFT method. Last but not least, FFT is very interesting for extensions of large even degree, which is not the case for the finite field $\mathbb{F}_{p^5}$. Consequently, we focused on the Newton interpolation.

4.2 Newton interpolation

Newton interpolation construct the polynomial $C(X)$ by the following way:

$$\begin{cases} c'_0 &= & C(\alpha_0) \\ c'_1 &= & (C(\alpha_1) - c'_0)\frac{1}{(\alpha_1-\alpha_0)} \\ c'_2 &= & \left((C(\alpha_2) - c'_0)\frac{1}{(\alpha_2-\alpha_0)} - c'_1\right)\frac{1}{(\alpha_2-\alpha_1} \\ \vdots &= & \vdots \\ C(X) = & c'_0 + c'_1(X-\alpha_0) + c'_2(X-\alpha_0)(X-\alpha_1) + \ldots + c'_{2k-2}(X-\alpha_0)(X-\alpha_1)\ldots(X-\alpha_{2k-2}) \end{cases}$$

The last line, corresponding to the interpolation, can be computed using the Horner scheme:

$$C(X) = c'_0 + (X-\alpha_0)\left[c'_1 + (X-\alpha_1)\left(c'_2 + (X-\alpha_2)\left\langle \ldots \right\rangle\right)\right]$$

So, complexity in term of operation of Newton interpolation is the sum of the complexity of this different operations:

1. the evaluations in α_i of $A(X)$ et $B(X)$
2. the $2k-1$ multiplications in $\mathbb{F}_p$ ($A(\alpha_i) \times B(\alpha_i)$)
3. computation of the c'_i
4. the Horner scheme to find the expression of $C(X) = A(X) \times B(X)$ of degree $2k-1$.

4.3 Simplifying operations in Newton interpolation for $k = 5$

We consider that $k = 5$, the $2k - 1 = 9$ chosen values for the interpolation are:

$$\alpha_0 = 0, \alpha_1 = 1, \alpha_2 = -1, \alpha_3 = 2, \alpha_4 = -2, \alpha_5 = 4, \alpha_6 = -4, \alpha_7 = 3, \alpha_8 = \infty.$$

We choose this value in order to minimize the number of additions and divisions by the differences of the α_i during the interpolation.

In the following section, we denote A_p an addition, M_p a multiplication, and S_p a square in $\mathbb{F}_p$.

Complexity of the evaluations in α_i of A and B First step, we have to evaluate $A(X)$ and $B(X)$ in the α_i's. With the chosen values, evaluations of $A(X)$ and $B(X)$ are done using only additions and shifts in $\mathbb{F}_p$. Indeed, a product by a power of 2 is composed of shifts in binary base, so to evaluate $A(X)$ in 2^j, we compute the products $a_i \times (2^j)^i$ which are shifts, and then the additions $\sum_{i=0}^{k-1} a_i (2^j)^i$.

Writing down $3 = 2 + 1$, the evaluation in 3 is only composed of shifts and additions too. Indeed, powers of 3 can be decomposed as sum of powers of 2: $3^2 = 2^3 + 1$, $3^3 = 2^5 - 2^2 - 1$ et $3^4 = 2^6 + 2^4 + 1$.

Adding the different costs, evaluations of $A(X)$ and $B(X)$ have a complexity of $74A_p$.

Once we have the evaluations, we have to compute the multiplications $A(\alpha_i) \times B(\alpha_i)$ which are obtain with $9M_p$.

The complexity of the step 1 and 2 is then $74A_p + 9M_p$.

Complexity of the computations of c'_j In order to compute the coefficients c'_j during a Newton interpolation, one has to compute exact divisions by the difference of the $\alpha_i \in \mathbb{F}_p$. We call an exact division a division where the dividend is a multiple of the divisor. Among all the differences of the α_i we choose, eleven are not a power of 2. They are described in Table 3. In a binary basis, exact divisions by power of 2 are very simple, they are only shift on the right of the bits. We have to analyze the complexity of this division by 3, 5 and 7. A precise analysis of these divisions gives the result that an exact division by 3, 5 or 7 can be computed in only one subtraction.

Table 3. The problematic differences

$e_3 - e_2 = 3$	$e_4 - e_1 = -3$	$e_5 - e_1 = 3$	$e_5 - e_2 = 5$
$e_5 - e_4 = 6$	$e_6 - e_1 = -5$	$e_6 - e_2 = -3$	$e_6 - e_3 = -6$
$e_7 - e_0 = 3$	$e_7 - e_4 = 5$	$e_7 - e_6 = 7$	

So we want to divide δ a multiple of 3 by 3, i.e. δ verifies that $\delta = 3 \times \sigma$ and we want to find σ. This equality can be rewritten as $\sigma = \delta - 2 \times \sigma$. If $\delta = \sum_i \delta_i 2^i$ and $\sigma = \sum_i \sigma_i 2^i$ we can find σ bit after bit beginning with the less significant bit. Indeed, $\sigma = \delta - 2 \times \sigma$ gives $\sigma_0 = \delta_0$. Thus we can find σ_1 as the result of the subtraction: $\delta_1\delta_0 - \sigma_0 0 = \sigma_1\sigma_0$. By extrapolation we find σ_1, and then σ_2 and the following as explained in Figure 4.3.

Consequently, an exact division by 3 is done with exactly one subtraction in $\mathbb{F}_p$. The same scheme can be applied to an exact division by 5. Indeed, for $\chi = 5 \times \kappa$ (i. e. knowing χ we want to find κ), we just have to consider that $\kappa = \chi - 4 \times \kappa$. Then the exact division by 5 has the complexity of a subtraction.

The complexity of an exact division by 7 is an addition in $\mathbb{F}_p$, provided that we find first the opposite of the result. We know $\mu = 7 \times \nu$, and we want to find ν. We transform the equation: $-\nu = \mu - 8 \times \nu$. So first we find $-\nu$ with an addition in $\mathbb{F}_p$, and then it is quite easy to find ν.

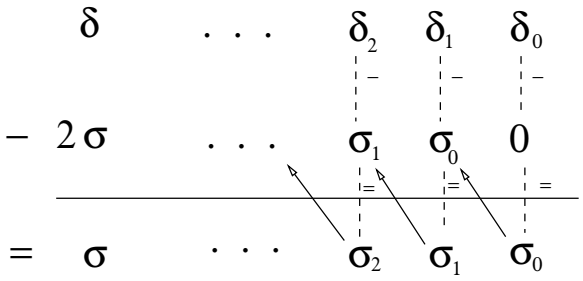

Fig. 1. Scheme for the division by 3 in one addition

We consider that the complexity of a subtraction is equivalent to the complexity of an addition, which is an upper bound for a subtraction. As a consequence, the exact divisions by 3, 5 and 7 can be computed with only an addition. The eleven divisions by these values have a complexity of $11A_p$.

In order to have the complexity of the computation of the c'_j, we must take in consideration the subtractions. There are 29 subtractions in the formulas of the c'_j.

Thus, the complexity of computing the c'_j is $40A_p$.

Complexity of the polynomial interpolation We use the Horner scheme to find the expression of the product polynomial $C = A \times B$. The Horner scheme consists in writing and computing:

$$C(X) = ((((c'_8(X - \alpha_7) + c'_7)(X - \alpha_6) + c'_6)(X - \alpha_5) + c'5) \ldots + c'_1)(X - \alpha_1) + c'0$$

We begin to compute from the inside (the parenthesis $(c'_8(X - \alpha_7) + c'_7)$) to the outside, i.e. we compute $((c'_8(X - \alpha_7) + c'_7)(X - \alpha_6) + c'_6)$, and we continue until we arrive at the coefficient c'_0. Thus the construction of the polynomial using the Horner scheme is composed of multiplications of the i^{th} parenthesis by α_{7-i} and additions. With the chosen values of α_i the Horner scheme is composed only of additions.

So the complexity of the polynomial expression with the Horner scheme is $29A_p$.

4.4 Results

We describe the multiplication by interpolation for an extension of degree 5 of a finite field. The Table 4 gives the complexity of a multiplication with several methods: the classical Karatsuba and Toom Cook multiplication (KTC), the interpolation multiplication, and the third line of Table 4 (Mix) gives the cost of a multiplication in $\mathbb{F}_{p^{15}}$ using an extension tower of field, we use our Newton multiplication in $\mathbb{F}_{p^5}$ and the Toom Cook multiplication in $\mathbb{F}_{p^{15}}$. We use interpolation for the intermediate field, and the Toom Cook multiplication for the final extension field.

Table 4. Complexity of different method of multiplication

Method \ Extension	M_{p^2}	M_{p^5}	$M_{p^{12}}$	$M_{p^{15}}$
KTC	$3M_p + 4A_p$	$13M_p + 58A_p$	$45M_p + 180A_p$	$65M_p + 390A_p$
Interpolation	$--$	$9M_p + 145A_p$	$23M_p + 2070A_p$	$29M_p + 2136A_p$
Mix	$--$	$--$	$45M_p + 180A_p$	$45M_p + 880A_p$

Using this result, for $e = 5$, it seems quite fair to use a multiplication by interpolation instead of a multiplication using Karatsuba Toom Cook. We save 4 multiplications in $\mathbb{F}_p$ using interpolation whereas we add 87 additions. The extra cost due to these additions is not as important as the cost to compute 4 multiplications in $\mathbb{F}_p$. Indeed, the complexity of a Karatsuba multiplication in $\mathbb{F}_p$ is

$$5N^{log_2(3)}A_{32} + N^{log_2(3)}M_{32},$$

where N is the number of bytes in the considered integers, and where A_{32} and M_{32} represent an addition and a multiplication of two words of size 32 bits. Considering an integer multiplication on a pentium 4, $M_{32} \approx 10A_{32}$[1]. Consequently Karatsuba multiplication of 20 bytes word has a complexity equivalent to $86A_p$, which is a lower bound of the complexity of a multiplication M_p in terms of additions A_p, because in this approximation we do not consider the reduction. Indeed, for a smart card, we have to take in consideration the reduction modulo p which adds additions in $\mathbb{F}_p$.

On the contrary, for extensions of degree 12 and 15, using an interpolation to compute a multiplication is not so interesting. The additional cost of the additions is huge in comparison to the saved multiplications. The interpolation method is very interesting for an extension of degree 5, because we can choose the value of interpolation such that the number of additions does not increase too much in relation with the saved multiplications. Table 5 gives the comparison of a pairing computation considering the number of multiplications and additions, at different security levels. The two last columns use the fact that for a pentium 4 $M_p \approx 86A_p$ using the Karatsuba multiplication. Comparing the equivalent number of multiplications for the pairing computation on curves with $k = 15$ and $k = 12$, we can conclude that our arithmetic gives better performances, we save at least 13% operations with our arithmetic. Indeed $M_p \approx 86A_p$ is a lower bound of the numbers of additions in $\mathbb{F}_p$ needed for a multiplication in $\mathbb{F}_p$.

Table 5. A performance evaluation: curves with embedding degree 15 versus Barreto-Naehrig curves

		twisted Ate		Pentium 4 equivalence in number of M_p	
AES security	bit length of r	k=15	k=12	k=15	k=12
80	160	$3075M_p + 48760A_p$	$4020M_p + 12960A_p$	4170	4641
128	256	$4920M_p + 78016A_p$	$6432M_p + 20736A_p$	5827	6673
192	384	$7380M_p + 117024A_p$	$9648M_p + 31104A_p$	8740	10009
256	512	$9840M_p + 156032A_p$	$12864M_p + 41472A_p$	11654	13346

The Table 6 gives our final comparison. We used our improved arithmetic for an extension of degree 5 with the Toom Cook method to compute a multiplication in $\mathbb{F}_{p^5}$. We compare our result with the complexity of the pairing on the Barreto-Naehrig curves. As we made an approximation for the number of addition, we compare the results using only the number of multiplications in $\mathbb{F}_p$ needed for a Miller loop. Moreover, in literature, only the number of multiplicatoin are considered. Our results show that we save 24% multiplications in $\mathbb{F}_p$ during the Miller algorithm for $k = 15$ compared to $k = 12$. We do not count the final exponentiation. The cost of the final exponentiation is obviously more expensive for the case $k = 15$ because it depends on the value of $\rho \approx 1,5$ and further work has to be done to find families of curves with a better ρ value.

[1] http://www.x86-secret.com/articles/cpu/p4/p4-6.htm

Table 6. A performance evaluation: overs curves with embedding degree 15 versus Barreto-Naehrig curves

AES security	bit length of r	twisted Ate	
		k=15	k=12
80	160	$3075M_p$	$4020M_p$
128	256	$4920M_p$	$6432M_p$
192	384	$7380M_p$	$9648M_p$
256	512	$9840M_p$	$12864M_p$

5 Conclusion

In this paper, we give efficient pairing computation for curves of embedding degree 15. We show that existing constructions of families of curves of degree 15 and j-invariant 0 present multiple advantages. First of all, we show that pairing computation on these curves has loop length $r^{1/8}$, which is an important gain when compared to $r^{1/4}$, which is the loop length of Miller's algorithm for curves of embedding degree 12. Secondly, we show that by using twists of degree 3 we manage to perform most of operations in F_p or F_{p^5}. Moreover, denominator computation and the final inversion can be avoided by making use of the twist. The application of the notion of optimal pairings to the Twisted Ate pairing is a real improvement. And thirdly, we present an efficient arithmetic in the finite field $\mathbb{F}_{p^5}$. Our results show that we save 24% multiplication for curves of embedding degree 15. Therefore curves of embedding degree 15 are competitive when compared to Barreto-Naehrig curves of embedding degree 12 and should thus be considered for cryptographic use. Moreover if we take in consideration the fact that the security level is going to increase, curves with higher embedding degree should be considered, even though at present for known families of such curves the ρ value is greater than the ideal value $\rho = 1$. Further research on building such curves might also improve this value.

References

1. S. Vanstone A. Menezes, T. Okamoto. Reducing elliptic curve logarithms in a finite field. *IEEE Transactions on Information Theory*, 39(5):1639–1646, 1993.
2. P. Barreto and M. Naehrig. Pairing-friendly elliptic curves of prime order. In *Selected Areas in Cryptography - SAC 2005*, volume 3897 of *Lecture Notes in Computer Science*, pages 319 –331. Springer, 2006.
3. D. Boneh and M. K. Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer Verlag, 2001.
4. D. Boneh, B. Lynn, and H. Shacham. Short signatures from the Weil pairing. In Colin Boyd, editor, *Advances in Cryptology – ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 514–532. Springer Verlag, 2001.
5. C. Cocks and R.G.E. Pinch. Indentity-based cryptosystems based on the Weil pairing. unplublished manuscript, 2001.
6. P. Duan, S. Cui, and C.W. Chan. Special polynomial families for generating more suitable elliptic curves for pairing-based cryptosystems. In *The 5th WSEAS International Conference on Electronics, Hardware, Wireless Optimal Communications*, 2005.
7. David Freeman, Michael Scott, and Edlyn Teske. A taxonomy of pairing-friendly elliptic curves. Cryptology ePrint Archive, Report 2006/372, 2006. http://eprint.iacr.org/.
8. Robert Granger, Dan Page, and Nigel P. Smart. High security pairing-based cryptography revisited. In Florian Hess, Sebastian Pauli, and Michael E. Pohst, editors, *ANTS VI*, volume 4076 of *Lectures Notes in Computer Science*, pages 480–494, 2006.
9. Florian Hess, Nigel P. Smart, and Frederik Vercauteren. The Eta Pairing Revisited. *IEEE Transactions on Information Theory*, 52:4595–4602, 2006.
10. A. Joux. A one round protocol for tripartite Diffie-Hellman. *Journal of Cryptology*, 17(4):263–276, September 2004.

11. Neal Koblitz and Alfred Menezes. Pairing-based cryptography at high security levels. In Nigel P. Smart, editor, *IMA Int. Conf.*, volume 3796 of *Lectures Notes in Computer Science*, pages 13–36, 2005.
12. X. Lin, C. Zhao, F. Zhang, and Y. Wang. Computing the Ate Pairing on Elliptic Curves with Embedding Degree k = 9. *IEICE Transactions*, 91-A(9):2387–2393, 2008.
13. Seiichi Matsuda, Naoki Kanayama, Florian Hess, and Eiji Okamoto. Optimised versions of the ate and twisted ate pairings. In *the Eleventh IMA International Conference on Cryptography and Coding*, pages 302–312. Springer-Verlag, 2007.
14. Victor S. Miller. The Weil pairing, and its efficient calculation. *Journal of Cryptology*, 17(4):235–261, September 2004.
15. H. Minkowski. *Geometrie der Zahlen*. Leipzig und Berlin, Druck und Verlag von B.G. Teubner, 1910.
16. Frederik Vercauteren. Optimal Pairings. http://eprint.iacr.org/2008/096, 2008.
17. André Weil. *Courbes algébriques et variétés abéliennes (in french.* Hermann, 1948.
18. J. Von ZurGathen and J. Gerhard. *Modern Computer Algebra*. Cambridge University Press, New York, NY, USA, 2003.