Implementing Cryptographic Pairings over Barreto-Naehrig Curves

Augusto Jun Devegili[1]*, Michael Scott[2], and Ricardo Dahab[1]

[1] Instituto de Computação, Universidade Estadual de Campinas
Caixa Postal 6176, CEP 13084-971 Campinas, SP, Brazil
augusto@devegili.org, rdahab@ic.unicamp.br
[2] School of Computing, Dublin City University
Dublin 9, Ireland
mscott@computing.dcu.ie

Abstract. In this paper we describe an efficient implementation of the Tate and Ate pairings using Barreto-Naehrig pairing-friendly curves, on both a standard 32-bit PC and on a 32-bit smartcard. First we introduce a sub-family of such curves with a particularly simple representation. Next we consider the issues that arise in the efficient implementation of field arithmetic in $\mathbb{F}_{p^{12}}$, which is crucial to good performance. Various optimisations are suggested, including a novel approach to the 'final exponentiation', which is faster and requires less memory than the methods previously recommended.

1 Introduction

Pairing-based cryptography requires pairing-friendly curves. These are parameterised by their embedding degree k. The embedding degree dictates to an extent the security level efficiently achievable on the curve.

While it is well known that super-singular curves are viable and useful candidates, they are limited in terms of the possible values of the embedding degree. Furthermore the highest embedding degree possible for super-singular elliptic curves ($k = 6$) requires us to use curves of characteristic 3, which is rather awkward from an implementation point of view (we refer the reader to a recent paper on arithmetic in GF(3^m) by Ahmadi, Hankerson and Menezes [1]). Attention has therefore switched to consideration of non-supersingular curves of prime characteristic, for which there is no such limitation. Nonetheless finding suitable curves, or ideally whole families of suitable curves, has proven to be non-trivial [2].

In this context the security of pairing-based cryptography depends on finding curves whose order n is divisible by a large prime r such that generic attacks on small group orders (Pohlig-Hellman attacks) can be resisted. It is also important that $k \lg(p)$, where p is the modulus, is large enough to resist index-calculus attacks.

* Funded by the Brazilian Government/Coordination for the Improvement of Higher Education Personnel (CAPES)

Given this scenario, Barreto-Naehrig (BN) curves with their embedding degree of 12 are particularly significant, as before their discovery only MNT curves [3] of embedding degree 3, 4 and 6 were known to support rare curves of prime order ($r = n$). The prime order requirement is significant in terms of efficiency of implementation [2], and is crucial for certain applications [4]. Furthermore since BN curves define a whole family of curves, there are plenty to choose from. One potential drawback of using BN curves is Schirokauer's proposed discrete logarithm algorithm [5], which may be applicable to BN primes because of their special format. In this case, it might be necessary to rescale parameters accordingly.

2 Bilinear Pairings

A bilinear pairing is a map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_3$ where $\mathbb{G}_1, \mathbb{G}_2$ are additive groups, $\mathbb{G}_3$ is a multiplicative group, and the map is linear in each component:

$$e(P+Q,R) = e(P,R) \cdot e(Q,R)$$
$$e(P,Q+R) = e(P,Q) \cdot e(P,R) .$$

Let $\mathbb{F}_p$ be the prime field with characteristic p and let $E(\mathbb{F}_p)$ be an elliptic curve defined over $\mathbb{F}_p$. Let n be the order of $E(\mathbb{F}_p)$, let r be a large prime dividing n, and let k be the least positive integer such that $r \mid p^k - 1$ and $r^2 \nmid p^k - 1$. We call such an integer k the embedding degree of r with regard to $\mathbb{F}_p$, and we know that the r-torsion group of the curve is contained in $E(\mathbb{F}_{p^k})$ and the r-th roots of unity are contained in $\mathbb{F}_{p^k}$.

Let $[a]P$ denote the multiplication of a point $P \in E$ by a scalar $a \in \mathbb{Z}$, and let $\infty \in E$ denote the point at infinity. A Miller [6] function $f_{r,P}(\cdot)$ is a rational function on E with r zeroes at P, one pole at $[r]P$ and $r-1$ poles at ∞:

$$(f_{r,P}) = r(P) - ([r]P) - (r-1)\infty .$$

The Tate pairing [7] is a well-defined, non-degenerate bilinear pairing with $\mathbb{G}_1 = E[r]$, $\mathbb{G}_2 = E(\mathbb{F}_{p^k})/rE(\mathbb{F}_{p^k})$, and $\mathbb{G}_3 = \mathbb{F}^*_{p^k}/(\mathbb{F}^*_{p^k})^r$. Let $P \in E[r]$ and $Q \in E(\mathbb{F}_{p^k})/rE(\mathbb{F}_{p^k})$. Then the Tate pairing of P, Q is computed as

$$e(P,Q) = f_{r,P}(Q)^{(p^k-1)/r} .$$

The Ate pairing [8] is a well-defined, non-degenerate bilinear pairing with $\mathbb{G}_1 = E[r] \cap \mathrm{Ker}(\pi_q - [1])$, $\mathbb{G}_2 = E[r] \cap \mathrm{Ker}(\pi_q - [q])$, and $\mathbb{G}_3 = \mathbb{F}^*_{p^k}/(\mathbb{F}^*_{p^k})^r$, where π_q is the Frobenius endomorphism. Let $P \in E[r] \cap \mathrm{Ker}(\pi_q - [1])$, $Q \in E[r] \cap \mathrm{Ker}(\pi_q - [q])$, and t be the trace of Frobenius of the curve. Then the Ate pairing of Q, P is computed as

$$e(Q,P) = f_{t-1,Q}(P)^{(p^k-1)/r} .$$

The most efficient pairing computation algorithms are derived from Miller's algorithm [6], which computes a Miller function $f_{r,P}$ evaluated at a point Q.

Algorithm 1 describes the BKLS algorithm [7] commonly used to compute the Tate pairing, including the denominator elimination optimisation that may be applied for even embedding degree. The algorithm needs $\mathbb{F}_{p^k}$ arithmetic (squaring and multiplication), elliptic curve arithmetic (point addition and doubling), computation of the line function $l_{A,B}(C)$ that intercepts points A, B and its evaluation at a point C, and a final exponentiation in $\mathbb{F}_{p^k}$.

Algorithm 1 BKLS algorithm to compute the Tate pairing $e(P, Q)$

```
Input: $P, Q \in E$ and $r \in \mathbb{Z}$
Output: $f_{r,P}(Q)^{(p^k-1)/r}$
 1: $T \leftarrow P$
 2: $f \leftarrow 1$
 3: for $i \leftarrow \lfloor \lg(r) \rfloor - 2$ downto $0$ do
 4:     $f \leftarrow f^2 \cdot l_{T,T}(Q)$
 5:     $T \leftarrow [2]T$
 6:     if $r_i = 1$ then
 7:         $f \leftarrow f \cdot l_{T,P}(Q)$
 8:         $T \leftarrow T + P$
 9:     end if
10: end for
11: $f \leftarrow f^{(p^k-1)/r}$
```

3 Barreto-Naehrig Curves

Barreto and Naehrig [9] devised a method to generate pairing-friendly elliptic curves over a prime field, with prime order and embedding degree $k = 12$. The equation of the curve is $E : y^2 = x^3 + b$, with $b \neq 0$. The trace (of Frobenius) of the curve, the curve order and the characteristic of $\mathbb{F}_p$ are parameterised as:

$$\begin{aligned} t(x) &= 6x^2 + 1 \\ n(x) &= 36x^4 - 36x^3 + 18x^2 - 6x + 1 \\ p(x) &= 36x^4 - 36x^3 + 24x^2 - 6x + 1\,. \end{aligned}$$

Such a curve is called a Barreto-Naehrig or BN curve. Throughout this text, we drop the parameter x and write t, n, p instead of $t(x)$, $n(x)$, $p(x)$ respectively. Because a BN curve has prime order, every point in the curve has order n, so the r value defined in Section 2 (a large prime dividing the curve order) is the same as n.

Since t, n, p are parameterised, the space needed to store or transmit information about a BN curve is small: The parameter x alone suffices. This parameterisation also allows for a faster final exponentiation method described in Section 7.

Because BN curves have embedding degree $k = 12$, pairings are computed over points in $E(\mathbb{F}_{p^{12}})$. A common optimisation [7] is to take one of the points in $E(\mathbb{F}_p) \subset E(\mathbb{F}_{p^{12}})$ and the other in $E(\mathbb{F}_{p^{12}})$. However, it is possible to do better than this: By using an appropriate map, we can compress certain points in $E(\mathbb{F}_{p^{12}})$ to points in a sextic twist $E'(\mathbb{F}_{p^2})$. Let $\xi \in \mathbb{F}_{p^2}$ be such that $W^6 - \xi$ is irreducible over $\mathbb{F}_{p^2}[W]$ whenever $p \equiv 1 \pmod 6$. Then there exists a curve $E'/\mathbb{F}_{p^2} : y'^2 = x'^3 + b/\xi$ that is a sextic twist of $E/\mathbb{F}_p$ and whose order is divisible by n. In fact, the order of the sextic twist $E'/\mathbb{F}_{p^2}$ is $n(2p - n)$.

Let $z \in \mathbb{F}_{p^{12}}$ be a root of $W^6 - \xi$. We can build an injective group homomorphism $\psi : E'(\mathbb{F}_{p^2}) \to E(\mathbb{F}_{p^{12}})$ as $(x', y') \mapsto (x'z^2, y'z^3)$. This allows us to map points in the sextic twist $E'(\mathbb{F}_{p^2})$ to points in $E(\mathbb{F}_{p^{12}})$, and if ξ is used to construct the extension field $\mathbb{F}_{p^{12}}$ then multiplication by powers of z does not incur any multiplication overhead.

We chose x such that p is a 256-bit prime subject to the following congruences: $p \equiv 7 \pmod 8$ (so that we can use -2 as a quadratic non-residue, optimising arithmetic operations on $\mathbb{F}_{p^2}$), $p \equiv 4 \pmod 9$ (as suggested in [9] to compute cube roots efficiently), and $p \equiv 1 \pmod 6$ (so that we can find $\xi \in \mathbb{F}_{p^2}$ such that $W^6 - \xi$ is irreducible over $\mathbb{F}_{p^2}[W]$). The elliptic curve equation is given by $E/\mathbb{F}_p : y^2 = x^3 + 3$, and $G = (1, 2)$ is a generator of $E(\mathbb{F}_p)$. For a 256-bit modulus, the x which defines the curve is just a 64-bit number.

Observe from Algorithm 1 that the Ate pairing benefits from r having low Hamming weight. In the context of BN curves, this is facilitated by preferring an x value of low Hamming weight. For example, we find that r has a Hamming weight of 90 and t has a Hamming weight of 28 if we choose $x =$ -6000000000001F2D (hex).

4 Finite Field Arithmetic

We construct the finite extension field $\mathbb{F}_{p^{12}}$ as a tower of finite extensions: Quadratic on top of a cubic on top of a quadratic. The quadratic/cubic non-residues and reduction polynomials are detailed in Table 1. The multiplication and squaring algorithms chosen to implement field arithmetic are listed in Table 2. We made the choice of multiplication and squaring algorithms based on the exhaustive testing described in [10].

Table 1. Extension fields

Extension	Non-Residue	Construction	Representation
$\mathbb{F}_{p^2}$	$\beta = -2$	$\mathbb{F}_p[X]/(X^2 - \beta)$	$a = a_0 + a_1 X$
$\mathbb{F}_{p^6}$	$\xi = -1 - \sqrt{\beta}$	$\mathbb{F}_{p^2}[Y]/(Y^3 - \xi)$	$a = a_0 + a_1 Y + a_2 Y^2$
$\mathbb{F}_{p^{12}}$	$\xi' = \sqrt[3]{\xi}$	$\mathbb{F}_{p^6}[Z]/(Z^2 - \xi')$	$a = a_0 + a_1 Z$

Table 2. Multiplication and squaring algorithms for finite extension fields

Extension	Multiplication	Squaring
$\mathbb{F}_p$	Comba	Comba
$\mathbb{F}_{p^2}$	Karatsuba	Complex
$\mathbb{F}_{p^6}$	Karatsuba	Chung-Hasan SQR2
$\mathbb{F}_{p^{12}}$	Karatsuba	Complex

Throughout this text we also use an alternative equivalent representation of elements in $\mathbb{F}_{p^{12}}$ based on z, a root of $(W^6 - \xi) \in \mathbb{F}_{p^2}[W]$:

$$\begin{aligned} a \in \mathbb{F}_{p^{12}} &= (a_{0,0} + a_{0,1}Y + a_{0,2}Y^2) + (a_{1,0} + a_{1,1}Y + a_{1,2}Y^2)Z \\ &= a_{0,0} + a_{1,0}z + a_{0,1}z^2 + a_{1,1}z^3 + a_{0,2}z^4 + a_{1,2}z^5 \,, \end{aligned}$$

where $a_{i,j} \in \mathbb{F}_{p^2}$.

5 The Tate Pairing

The Tate pairing $e(P, Q)$ takes a point $P = (x_P, y_P) \in E(\mathbb{F}_p)$ and a point $Q = (x_Q, y_Q) \in E(\mathbb{F}_{p^{12}})$. Recall from Section 3 that BN curves have sextic twists defined over $\mathbb{F}_{p^2}$, so we can use a point $Q' = (x'_Q, y'_Q) \in E'(\mathbb{F}_{p^2})$ instead of the full point $Q \in E(\mathbb{F}_{p^{12}})$, and that there exists a group homomorphism $\psi : E'(\mathbb{F}_{p^2}) \to E(\mathbb{F}_{p^{12}})$ defined as $\psi((x', y')) \mapsto (x'z^2, y'z^3)$. The coordinates of a point Q in the codomain of ψ have a compact representation, needing at most 1/6 of the number of $\mathbb{F}_p$ elements needed to represent a full $\mathbb{F}_{p^{12}}$ element.

In Algorithm 1, the point P is repeatedly doubled or added during the Miller loop, and therefore we need elliptic curve arithmetic on $E(\mathbb{F}_p)$.

We use $\lambda_{A,B}$ in order to evaluate $l_{A,B}(Q)$. Because $A, B \in E(\mathbb{F}_p)$, every coordinate is an element of $\mathbb{F}_p$ and thus $\lambda_{A,B} \in \mathbb{F}_p$. Let $C = A + B$. Using projective coordinates,

$$\begin{aligned} l_{A,B}(Q) &= (y_Q \cdot z_A^3 - y_A)z_C - (x_Q \cdot z_A^3 - x_A \cdot z_A)\lambda_{A,B} \\ &= y'_Q \cdot z^3 \cdot z_A^3 \cdot z_C - y_A \cdot z_C - x'_Q \cdot z^2 \cdot z_A^3 \cdot \lambda_{A,B} - x_A \cdot z_A \cdot \lambda_{A,B} \\ &= (x_A \cdot z_A \cdot \lambda_{A,B} - y_A \cdot z_C) - (x'_Q \cdot z_A^3 \cdot \lambda_{A,B})z^2 + (y'_Q \cdot z_A^3 \cdot z_C)z^3 \,. \end{aligned}$$

Since $x_A, y_A, \lambda_{A,B} \in \mathbb{F}_p$ and $x'_Q, y'_Q \in \mathbb{F}_{p^2}$, we can avoid $\mathbb{F}_{p^{12}}$ arithmetic entirely when computing $l_{A,B}(Q)$.

6 The Ate Pairing

The Ate pairing $e(Q, P)$ takes a point $Q = (x_Q, y_Q) \in E(\mathbb{F}_{p^{12}})$ and a point $P = (x_P, y_P) \in E(\mathbb{F}_p)$. As in Section 5, we can use a point $Q' = (x'_Q, y'_Q) \in E'(\mathbb{F}_{p^2})$ instead of the full point $Q \in E(\mathbb{F}_{p^{12}})$.

The Miller loop needs two elliptic curve arithmetic operations: Point doubling and point addition. As these operations are computed over Q', we need elliptic curve arithmetic over $E'(\mathbb{F}_{p^2})$. Instead of using r to control the Miller loop, the Ate pairing uses $s = t - 1$, which in the case of BN curves provides for roughly half the number of iterations needed to compute the Tate pairing. Algorithm 2 describes the BKLS algorithm adapted to compute the Ate pairing.

Algorithm 2 BKLS-like algorithm to compute the Ate pairing $e(Q, P)$

```
Input: $P, Q \in E$ and $r, s = t - 1 \in \mathbb{Z}$
Output: $f_{t-1,Q}(P)^{(p^k-1)/r}$
 1: $T \leftarrow Q$
 2: $f \leftarrow 1$
 3: for $i \leftarrow \lfloor \lg(s) \rfloor - 2$ downto $0$ do
 4:     $f \leftarrow f^2 \cdot l_{T,T}(P)$
 5:     $T \leftarrow [2]T$
 6:     if $s_i = 1$ then
 7:         $f \leftarrow f \cdot l_{T,Q}(P)$
 8:         $T \leftarrow T + Q$
 9:     end if
10: end for
11: $f \leftarrow f^{(p^k-1)/r}$
```

The formula for the slope of the line function $l_{A,B}$ that intercepts two points $A = (x_A, y_A)$ and $B = (x_B, y_B)$ is

$$\lambda_{A,B} = \frac{y_B - y_A}{x_B - x_A} .$$

If $A, B \in E(\mathbb{F}_{p^{12}})$, the slope is an element of $\mathbb{F}_{p^{12}}$. We can easily derive this from the slope of the line function $l_{A',B'}$ that intercepts $A', B' \in E'(\mathbb{F}_{p^2})$, where A', B' are the images of A, B under ψ:

$$\lambda_{A,B} = \frac{(y'_B - y'_A)z^3}{(x'_B - x'_A)z^2} = (\lambda_{A',B'})z ,$$

so computing $\lambda_{A,B} \in \mathbb{F}_{p^{12}}$ amounts to computing $\lambda_{A',B'} \in \mathbb{F}_{p^2}$. We use this slope to evaluate $l_{A,B}(P)$ as follows:

$$\begin{aligned} l_{A,B}(P) &= (x_P - x_A)\lambda_{A,B} - (y_P - y_A) \\ &= (x_P - x'_A \cdot z^2)(\lambda_{A',B'})z - (y_P - y'_A \cdot z^3) \\ &= (-y_P) + (x_P \cdot \lambda_{A',B'})z + (y'_A - x'_A \cdot \lambda_{A',B'})z^3 . \end{aligned}$$

In practise the use of projective coordinates for $E(\mathbb{F}_{p^2})$ point doublings and additions is a little faster, even though for the pairing calculation elliptic curve operations take up relatively little time compared with the $\mathbb{F}_{p^{12}}$ operations.

As $x_P, y_P \in \mathbb{F}_p$ and $x'_A, y'_A, \lambda_{A',B'} \in \mathbb{F}_{p^2}$, again we can avoid $\mathbb{F}_{p^{12}}$ arithmetic entirely when computing $l_{A,B}(P)$. The resulting value has a sparse representation in $\mathbb{F}_{p^{12}}$, needing only five $\mathbb{F}_p$ elements instead of twelve. However, the Miller variable is a full $\mathbb{F}_{p^{12}}$ element. For a given point P, the value $-y_P$ is constant, so it needs to be computed only once during the pairing computation.

7 Final Exponentiation

Both the Ate and Tate pairing algorithms compute a final exponentiation after the Miller loop. A common optimisation of this computation is to factor $(p^k - 1)/n$ into three parts: An easy exponentiation to the power of $(p^{k/2} - 1)$, an equally easy exponentiation to the power of $(p^{k/2} + 1)/\Phi_k(p)$ (easy because of the Frobenius), and a 'hard' exponentiation to the power of $\Phi_k(p)/n$, where Φ_k is the k-th cyclotomic polynomial [11]. In the context of BN curves, these exponents translate to $(p^6 - 1)$, $(p^2 + 1)$, and $(p^4 - p^2 + 1)/n$.

Recall that p and n have a special form: Both are polynomials on x. Therefore this hard part of the final exponentiation can be computed explicitly as a large polynomial in x. This can in turn be expressed to the base p as

$$p^3 + (6x^2 + 1)p^2 + (36x^3 - 18x^2 + 12x + 1)p + (36x^3 - 30x^2 + 18x - 1)\,.$$

Now the standard continuation is to use the method of multi-exponentiation combined with the Frobenius [12], so that the final exponentiation is the calculation of

$$(f^{p^3}) \cdot (f^{p^2})^{6x^2+1} \cdot (f^p)^{36x^3-18x^2+12x+1} \cdot f^{36x^3-30x^2+18x-1}\,.$$

However, multi-exponentiation is expensive in terms of memory as it requires extensive precomputation. So instead we exploit the specific and fixed form of the final exponent to obtain Algorithm 3 which can easily be verified to produce the equivalent result, but only requires simple exponentiation.

Algorithm 3 'Hard' exponentiation

```
Input: $f, x, p$
Output: $f^{(p^4-p^2+1)/n}$
1: $a \leftarrow f^{6x-5}$
2: $b \leftarrow a^p$ using Frobenius
3: $b \leftarrow ab$
4: Compute $f^p$, $f^{p^2}$, and $f^{p^3}$ using Frobenius
5: $f \leftarrow f^{p^3} \cdot [b \cdot (f^p)^2 \cdot f^{p^2}]^{6x^2+1} \cdot b \cdot (f^p \cdot f)^9 \cdot a \cdot f^4$
```

Note that exponentiations to powers of p are efficiently computed using Frobenius; other exponentiations may be computed using the square-and-multiply method. Because x is chosen so as to have low Hamming weight, there is no benefit in using window methods to compute the exponentiations to the powers of $6x - 5$ and $6x^2 + 1$, which saves on memory. Experiments show this method to be about 20% faster than using multi-exponentiation.

8 The Philips HiPerSmart$^{\text{TM}}$ Smartcard

As in [13], we used the Philips HiPerSmart$^{\text{TM}}$ smartcard, which is an instantiation of the MIPS32®-based SmartMIPS® architecture with various instruction set enhancements to facilitate the implementation of popular cryptographic algorithms. Fortunately these enhancements are also relevant to our efforts here.

Specifically the architecture supports a `ACX|HI|LO` triple of registers that can be used to accumulate the partial products that arise when employing the popular Comba/Montgomery technique for multi-precision multiplication [14]. This is supported by a modified `MADDU` instruction which carries out an unsigned integer multiplication and addition to the triple register.

One architectural feature of particular significance is the 2 KB instruction cache. This is appropriate for an algorithm which spends most of its time in a small inner loop, as say for a modular exponentiation as required by the RSA algorithm. However it is problematical for the more complex algorithms considered here. Therefore we found that it was hard to avoid frequent cache misses, which are particularly expensive at the higher clock speeds. In particular it was self-defeating to try and use the popular optimisation of loop-unrolling, as the reduction in instruction count was more than offset by an increase in clock cycle count.

By using stack allocation for the multiprecision variables it was possible to keep the RAM requirement within the 16 KB allocation. However this could have been a problem if we were to use methods which are more memory-hungry (like multi-exponentiation).

9 Results

The performance of the Ate and Tate pairings was measured on two platforms: The Philips HiPerSmart$^{\mathrm{TM}}$ platform described in Section 8, and a 32-bit Intel Pentium IV 3.0 GHz (Prescott) processor.

Our programs used the MIRACL library[3], which implements multi-precision number arithmetic, and supports a number of powerful optional optimizations. In particular it supports completely unrolled assembly language support for fixed-size big number multiplication and modular reduction. Internally, prime field elements are in Montgomery representation [15], which allows for fast reduction without divisions. The memory for big numbers can be allocated from the heap, or more efficiently from the stack, which is the case for our experiments. When required to multiply big numbers by a small integer, multiplications by numbers less than or equal to 6 are instead carried out by up to 3 modular additions.

Table 3 lists the number of instructions required to compute the pairings on the smartcard, as well as the number of cache misses. The count of clock cycles, the average Clocks Per Instruction (CPI), and the actual time needed to compute the pairing is listed in Tables 4 and 5. The timings for the 3.0 GHz Pentium IV are listed in Table 6. Here the implementation takes full advantage of the Pentium's SSE2 instruction set enhancements.

Our smartcard hardware emulator is only cycle-accurate up to 20.57 MHz, but the maximum supported speed for the smartcard is 36 MHz. At this clock

[3] `http://www.shamus.ie`

Table 3. Instructions required (% icache misses) — Philips HiPerSmart$^{\text{TM}}$

	Ate pairing	Tate pairing
Miller loop	37,929,400 (23.5%)	65,027,846 (20.5%)
Final exponentiation	22,161,919 (23.9%)	22,170,594 (23.2%)
Total	60,091,319 (23.7%)	87,198,440 (21.2%)

Table 4. Clock cycles required/CPI/time in seconds @ 9 MHz

	Ate pairing	Tate pairing
Miller loop	57,013,051/1.50/6.33	94,496,817/1.45/10.50
Final exponentiation	33,448,982/1.51/3.72	33,319,017/1.50/3.70
Total	90,462,033/1.51/10.05	127,815,834/1.47/14.20

frequency the Ate pairing over BN curves should take approximately 3 seconds, which is not adequate for use at the moment. Nevertheless it is anticipated that, with improvements in technology, BN curves might be practical even on such resource constrained devices in the near future.

10 Conclusions

We have described the first implementation of both the Ate and Tate pairings over Barreto-Naehrig curves.

Because of the restrictions imposed on the selection of primes for BN curves, we have not used primes congruent to 1 (mod 12), and therefore our finite fields are not strictly pairing-friendly in the sense of Koblitz and Menezes [16]. We have used the technique of towering extensions to construct the extension field $\mathbb{F}_{p^{12}}$, and our choice of non-residues for the reduction polynomials that define the extensions still allows for efficient finite field arithmetic.

In order to avoid full $\mathbb{F}_{p^{12}}$ arithmetic throughout the pairing computation, we have provided explicit $\mathbb{F}_{p^2}$-formulae for the evaluation of the line function $l_{A,B}(C)$ required by the Miller algorithm, thus alleviating part of the burden imposed by using the $k = 12$ embedding degree. The value of $l_{A,B}(C)$ computed by the Ate (resp. Tate) pairing uses five (resp. six) $\mathbb{F}_p$ components instead of twelve.

Our method for the final exponentiation is both faster and less memory-intensive than previous methods in the literature which are based on multi-exponentiation. We also observe that the analysis of Hess, Smart and Ver-

Table 5. Clock cycles required/CPI/time in seconds @ 20.57 MHz

	Ate pairing	Tate pairing
Miller loop	67,018,425/1.77/3.26	109,697,928/1.69/5.33
Final exponentiation	39,379,515/1.78/1.91	39,139,099/1.77/1.90
Total	106,397,940/1.77/5.17	148,837,027/1.71/7.24

Table 6. Timings in milliseconds on a 32-bit 3.0 GHz Intel Pentium IV

	Ate pairing	Tate pairing
Miller loop	14.8	25.4
Final exponentiation	8.4	8.4
Total	23.2	33.8

cauteren [8] for the settings of BN curves ($k = 12$, $\lg(p) = \lg(r) = 256$) is more optimistic with regard to the Ate pairing than our experimental results.

Acknowledgements We would like to thank the anonymous reviewers for their valuable comments, and Décio Gazzoni for insightful information on the number field sieve. Thanks also to Alfred Menezes and Darrel Hankerson for pointing out a problem with our results as originally reported.

References

1. Omran Ahmadi, Darrel Hankerson, and Alfred Menezes. Software implementation of arithmetic in GF(3^m). In *WAIFI 2007 (to be published)*, 2007.
2. 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/`.
3. Atsuko Miyaji, Masaki Nakabayashi, and Sunzou Takano. New explicit conditions of elliptic curve traces for FR-reduction. *IEICE Trans. Fundamentals*, E84-A(5):1234–1243, May 2001.
4. D. Boneh, B. Lynn, and H. Schacham. Short signatures from the Weil pairing. *Journal of Cryptology*, 17(4):297–319, 2004.
5. Oliver Schirokauer. The number field sieve for integers of low weight. Cryptology ePrint Archive, Report 2006/107, 2006. `http://eprint.iacr.org/`.
6. Victor S. Miller. The Weil pairing, and its efficient calculation. *Journal of Cryptology*, 17(4):235–261, September 2004.
7. Paulo S. L. M. Barreto, Hae Y. Kim, Ben Lynn, and Michael Scott. Efficient algorithms for pairing-based cryptosystems. In *CRYPTO 2002*, number 2442 in Lecture Notes in Computer Science, pages 354–369, Berlin Heidelberg, 2002. Springer-Verlag.
8. Florian Hess, Nigel P. Smart, and Frederik Vercauteren. The Eta Pairing Revisited. *IEEE Transactions on Information Theory*, 52(10):4595–4602, October 2006.
9. Paulo S. L. M. Barreto and Michael Naehrig. Pairing-friendly elliptic curves of prime order. In B. Preneel and S. Tavares, editors, *SAC 2005*, number 3897 in Lecture Notes in Computer Science, pages 319–331, Berlin Heidelberg, 2006. Springer-Verlag.
10. Augusto Jun Devegili, Colm Ó hÉigeartaigh, Michael Scott, and Ricardo Dahab. Multiplication and squaring on pairing-friendly fields. Cryptology ePrint Archive, Report 2006/471, 2006. `http://eprint.iacr.org/`.
11. 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*, volume 4076 of *Lecture Notes in Computer Science*, pages 480–494. Springer, 2006.

12. Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography.* CRC Press, Boca Raton, 1996.
13. Michael Scott, Neil Costigan, and Wesam Abdulwahab. Implementing cryptographic pairings on smartcards. In L. Goubin and M. Matsui, editors, *CHES 2006*, volume 4249 of *Lecture Notes in Computer Science*, pages 134–147, 2006.
14. Johann Großschädl and Erkay Savas. Instruction set extensions for fast arithmetic in finite fields GF(p) and GF(2^m). In Marc Joye and Jean-Jacques Quisquater, editors, *CHES 2004*, volume 3156 of *Lecture Notes in Computer Science*, pages 133–147. Springer, 2004.
15. Peter L. Montgomery. Modular multiplication without trial division. *Mathematics of Computation*, 44(170):519–521, April 1985.
16. Neal Koblitz and Alfred Menezes. Pairing-based cryptography at high security levels. In N.P. Smart, editor, *Cryptography and Coding 2005*, volume 3796 of *Lecture Notes in Computer Science*, pages 13–36, Berlin Heidelberg, 2005. Springer-Verlag.