

Implementing cryptographic pairings at standard security levels

Andreas Enge, Jérôme Milan

▶ **To cite this version:**

Andreas Enge, Jérôme Milan. Implementing cryptographic pairings at standard security levels. Security, Privacy, and Applied Cryptography Engineering, Oct 2014, Pune, India. pp.28-46, 10.1007/978-3-319-12060-7_3 . hal-01034213

HAL Id: hal-01034213
https://inria.hal.science/hal-01034213

Submitted on 22 Jul 2014

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Implementing cryptographic pairings at standard security levels

Andreas Enge* and Jérôme Milan†

22 July 2014

Abstract

This study reports on an implementation of cryptographic pairings in a general purpose computer algebra system. For security levels equivalent to the different AES flavours, we exhibit suitable curves in parametric families and show that optimal ate and twisted ate pairings exist and can be efficiently evaluated. We provide a correct description of Miller's algorithm for signed binary expansions such as the NAF and extend a recent variant due to Boxall *et al.* to addition-subtraction chains. We analyse and compare several algorithms proposed in the literature for the final exponentiation. Finally, we give recommendations on which curve and pairing to choose at each security level.

1 Pairings on elliptic curves

In this article, we treat cryptographic bilinear pairings $G_1 \times G_2 \to G_T$ on elliptic curves E defined over some finite field $\mathbb{F}_q$ of characteristic p. We emphasise that our aim is not to set new speed records for particular curves, cf. [2, 20, 1], but to compare different choices of pairings and parameters at various security levels, using a general purpose, but reasonably optimised, implementation in a general purpose computer algebra system. Such an analysis will be meaningful assuming that the ratios between the various operations remain constant when switching to hand-optimised assembly implementations in each instance.

We fix the following standard notations and setting. Let $E(\mathbb{F}_q)$ denote the $\mathbb{F}_q$-rational points on E, and let r be a prime divisor of $\#E(\mathbb{F}_q) = q + 1 - t$ that does not divide $q - 1$, where t is the trace of Frobenius. Let the embedding degree k be the smallest integer such that r divides $q^k - 1$, and denote by π the Frobenius map $E(\mathbb{F}_{q^k}) \to E(\mathbb{F}_{q^k})$, $(x, y) \mapsto (x^q, y^q)$. The r-torsion subgroup $E[r]$ is defined over $\mathbb{F}_{q^k}$, and it contains the non-trivial subgroup $E(\mathbb{F}_q)[r]$ of $\mathbb{F}_q$-rational r-torsion points. Denote by μ_r the subgroup of r-th roots of unity in $\mathbb{F}_{q^k}^*$.

Typically, $G_T = \mu_r$, $G_1 = E(\mathbb{F}_q)[r]$, and G_2 is a subgroup of order r of either $E[r]$ or of $E(\mathbb{F}_{q^k})/rE(\mathbb{F}_{q^k})$.

1.1 Functions with prescribed divisors

Let E be given over $\mathbb{F}_q$ by an equation in the variables x and y. For a rational function $f \in \overline{\mathbb{F}}_q(E) := \overline{\mathbb{F}}_q(x)[y]/(E)$ and a point $P \in E$, denote by $\mathrm{ord}_P(f)$ the positive multiplicity of the

*INRIA, LFANT, F-33400 Talence, France
CNRS, IMB, UMR 5251, F-33400 Talence, France
Univ. Bordeaux, IMB, UMR 5251, F-33400 Talence, France
andreas.enge@inria.fr
†INRIA, CNRS, UMR 7161, École polytechnique, LIX, 91128 Palaiseau, France

zero P of f, the negative multiplicity of the pole P of f, or 0 if P is neither a zero nor a pole of f. Denote by $\mathrm{div}(f) = \sum_P \mathrm{ord}_P(f)[P]$ the divisor of f, an element of the free abelian group generated by the symbols $[P]$, where P is a point on E.

The definition and computation of pairings involve certain rational functions with given divisors, in particular, $f_{n,P}$ with

$$\mathrm{div}(f_{n,P}) = n[P] - [nP] - (n-1)[\mathcal{O}],$$

the lines $\ell_{P,Q}$ through two (not necessarily distinct) points P and Q with

$$\mathrm{div}(\ell_{P,Q}) = [P] + [Q] + [-(P+Q)] - 3[\mathcal{O}]$$

and the vertical lines v_P through a point P with

$$\mathrm{div}(v_P) = [P] + [-P] - 2[\mathcal{O}].$$

All these functions are defined up to a multiplicative constant, and they are normalised at infinity by the condition $\left(f\left(\frac{Y}{X}\right)^{\mathrm{ord}_{\mathcal{O}}(f)}\right)(\mathcal{O}) = 1$.

In particular, we have $\ell_{P,-P} = v_P$, $f_{1,P} = 1$ and $f_{-1,P} = 1/v_P$.

The function $f_{n,P}$ is of degree $O(n)$ and may be evaluated in $O(\log n)$ steps by the algorithms of §3.

1.2 Cryptographic pairings

We quickly recall the main cryptographic pairings. In applications, they are usually restricted to $E(\mathbb{F}_q)[r]$ in one argument and to a subgroup of order r in the other argument.

Weil pairing

$$e_{\mathrm{W}} : E(r) \times E(r) \to \mu_r, \quad (P,Q) \mapsto (-1)^r \frac{f_{r,P}(Q)}{f_{r,Q}(P)}$$

Computing the pairing requires the evaluation of two functions; moreover, with $P \in E(\mathbb{F}_q)$ and $Q \in E(\mathbb{F}_{q^k})$, the function $f_{r,Q}$ is much costlier to evaluate by the algorithms of §3.

Tate pairing

$$\begin{array}{ccccc} e_{\mathrm{T}} : E(\mathbb{F}_q)[r] \times E(\mathbb{F}_{q^k})/rE(\mathbb{F}_{q^k}) & \to & \mathbb{F}_{q^k}^*/(\mathbb{F}_{q^k}^*)^r & \simeq & \mu_r, \\ (P,Q) & \mapsto & f_{r,P}(Q) & \leftrightarrow & f_{r,P}(Q)^{(q^k-1)/r}. \end{array}$$

The pairing requires only one evaluation of a rational function, but the original definition with a quotient group as domain is unwieldy since there is no easy way of defining unique representatives. The final exponentiation step of raising to the power $\frac{q^k-1}{r}$ realises an isomorphism with μ_r, and the resulting pairing is usually called the reduced Tate pairing.

Ate pairing By restricting its arguments to the two eigenspaces of $E[r]$ under the Frobenius with eigenvalues 1 and q, respectively, the ate pairing introduced in [14] replaces $f_{r,P}(Q)$ with $r \in O(q)$ by $f_{T,Q}(P)$, where $T = t - 1 \in O(\sqrt{q})$. This saving may be offset by the swap of arguments P and Q, so that the function is defined over $\mathbb{F}_{q^k}$ instead of $\mathbb{F}_q$.

$$e_{\mathrm{A}} : E(\mathbb{F}_q)[r] \times E[r] \cap \ker(\pi - q) \to \mu_r, \quad (P,Q) \mapsto f_{T,Q}(P)^{(q^k-1)/r}.$$

Twisted ate pairing The curve E admits a twist of degree $d = \gcd(k, \#\operatorname{Aut}(E))$; let $e = k/d$. The twisted ate pairing of [14] works again with a function over $\mathbb{F}_q$, at the expense of a factor of $O(e)$ in its evaluation:

$$e_{\mathrm{A}'} : E(\mathbb{F}_q)[r] \times E[r] \cap \ker(\pi - q) \to \mu_r, \quad (P, Q) \mapsto f_{T^e,P}(Q)^{(q^k-1)/r}.$$

Optimal pairing Generalisations of the ate and twisted ate pairings requiring several functions for their evaluation have been given in [13, 23]. All of them take $G_1 = E(\mathbb{F}_q)$ and $G_2 = E[r] \cap \ker(\pi - q)$. They are evaluated with low degree functions, typically requiring $O(\log r/\varphi(k))$ operations (not counting the final exponentiation), where φ is Euler's function.

Let $\lambda = mr = \sum_{i=0}^{n} \lambda_i q^i$ be a suitably chosen multiple of r with $r \nmid m$ such that the λ_i are small; more precisely, one requires a short addition-subtraction sequence passing through all λ_i. An optimal ate pairing is obtained by

$$e_{\mathrm{O}} : G_1 \times G_2 \to \mu_r, \quad (P, Q) \mapsto \left(\prod_{i=0}^{n} f_{\lambda_i,Q}^{q^i}(P) \prod_{i=0}^{n-1} \frac{\ell_{s_{i+1}Q,\lambda_i q^i Q}(P)}{v_{s_i Q}(P)} \right)^{(q^k-1)/r},$$

where $s_i = \sum_{j=i}^{n} \lambda_j q^j$. Since $\lambda = \varphi_k(q)$ yields a degenerate pairing, one may assume $n < \varphi(k)$; a precise condition for non-degeneracy is given in [23, Th. 4]. Finding such a multiple of r with small coefficients in base q is a common integer relation problem. It may be solved, for example, by using the LLL algorithm to find a short vector $(\lambda_0, \ldots, \lambda_{\varphi(k)-1})$ in the lattice generated by $(r, 0, \ldots, 0)$ and the $(-q^{i-1}, 0, \ldots, 0, 1, 0, \ldots, 0)$ with 1 at position i for $i = 2, \ldots, \varphi(k)$.

Optimal twisted pairing In the presence of a twist of degree d such that $k = de$, a pairing can be obtained in a similar fashion from $\lambda = mr = \sum_{i=0}^{n} \lambda_i T^{ei}$ with $n < \varphi(d)$. The only interesting cases are $d \in \{3, 4, 6\}$ with $\varphi(d) = 2$ (otherwise, we obtain again the twisted ate pairing). Then

$$e_{\mathrm{O}'} : G_1 \times G_2 \to \mu_r, \quad (P, Q) \mapsto \left(f_{\lambda_0,P}(Q) f_{\lambda_1,P}^{q^e}(Q) v_{\lambda_0 P}(Q) \right)^{(q^k-1)/r}$$

defines a pairing, where (λ_0, λ_1) is a short vector in the lattice generated by $(r, 0)$ and $(-T^e, 1)$.

2 Curves and associated optimal pairings

2.1 Curve selection

Pairing-based cryptographic settings should be designed so that the discrete logarithm problem is intractable in each group involved. Current best attacks on a subgroup of prime order r of an elliptic curve require on the order of $\sqrt{r}$ operations. For the sake of efficiency of the implementation and to save bandwidth in transmitting points on the curve, it would be desirable that $\#E(\mathbb{F}_q) = r \approx q$ by Hasse's theorem. This condition, however, is not easily met for arbitrary embedding degree k, and the parameter $\rho = \frac{\log q}{\log r}$ measures to which extent it is violated: Values close to 1 would be optimal. Discrete logarithms in finite fields $\mathbb{F}_{q^k}$ of large characteristic, which we deal with in this article, may be computed by algorithms with subexponential complexity $L_{q^k}(1/3)$ (as opposed to quasi-polynomial complexity in small characteristic [5]). One is thus looking for parameter values such that $\sqrt{r} \approx L_{q^k}(1/3) = L_{r^{\rho k}}(1/3)$. Taking logarithms on both sides shows that for bounded ρ, the embedding degree k grows asymptotically as $\log^2 q$.

Security	$\log_2 r$	$\log_2 q^k$	Target $k\rho$
128	256	3248	12.7
192	384	7936	20.7
256	512	15424	30.1

Table 1: Recommended curve sizes

Several studies have refined this argument. Table 1 summarises security levels equivalent to the AES secret key sizes, all given in bits, according to the Ecrypt2 recommendations [22], validated by a large community of cryptologists.

For each security level, we have selected from [9] several curve families approximately fulfilling the requirements on the parameter sizes, trying to stay as close as possible to the ideal $k\rho$ values.

These curves, together with their main parameters, are given in Table 2; for more details, see Appendix A. The entries in column "construction" refer to the algorithms of [9]. Supersingular curves are ruled out at these security levels by their too small embedding degree $k \leq 6$, so we restricted the search to ordinary curves. Since non-prime base fields $\mathbb{F}_q$ are virtually impossible to reach, all curves are defined over prime fields $\mathbb{F}_p$. We favoured a small Hamming weight of r and field extensions $\mathbb{F}_{p^k}$ which may be defined by a binomial $X^k - a$ with a small $a \in \mathbb{F}_p$. For comparison purposes, we also included the Barreto-Naehrig curves $E_{12,2}$ and $E_{12,3}$, widely considered the best choice for 128 bit security due to their optimal value of $\rho = 1$, for security levels of 192 and 256 bits by artificially increasing p. Some curve families are very sparse, which forced us to relax the constraints, for instance for E_{25}.

Security	k	ρ	$k\rho$	Curve	Construction	$\deg q$	$\deg r$	$\deg t$	$\varphi(k)$
128	9	4/3	12.0	E_9	6.6	8	6	4	6
	11	6/5	13.2	E_{11}	6.6	24	20	12	10
	12	1	12	$E_{12,1}$	6.8	4	4	2	4
192	12	1	12	$E_{12,2}$	6.8	4	4	2	4
	15	3/2	22.5	E_{15}	6.6	12	8	6	8
	16	5/4	20	E_{16}	6.11	10	8	5	8
	17	19/16	20.2	E_{17}	6.2	38	32	2	16
	18	4/3	24	E_{18}	6.12	8	6	4	6
	19	10/9	21.1	E_{19}	6.6	40	36	20	18
256	12	1	12	$E_{12,3}$	6.8	4	4	2	4
	24	5/4	30	E_{24}	6.6	10	8	1	8
	25	13/10	32.5	E_{25}	6.6	52	40	26	20
	26	7/6	30.34	E_{26}	6.6	28	24	14	12
	27	10/9	30	E_{27}	6.6	20	18	10	18

Table 2: Main curve parameters

2.2 Optimal pairings

Each curve family gives rise to different optimal pairings, which are obtained via short vectors in certain lattices as explained in §1.2. For each curve family in Table 2, such short vectors are given in Table 11. Notice that for E_{17} and E_{24}, the ate pairings are already optimal, and that for $d \notin \{3, 4, 6\}$ and moreover for E_{24}, the twisted optimal pairings are exactly the twisted ate pairings.

3 The Miller loop

The procedure to evaluate the function $f_{n,P}$ is given by Miller in [17] and relies on the recursive relations $f_{1,P} = 1$ and

$$f_{n+m,P} = f_{n,P} f_{m,P} \frac{\ell_{nP,mP}}{v_{(n+m)P}}, \tag{1}$$

which can be easily checked by taking divisors on both sides of the equality, see §1.1. This allows to compute $f_{x,P}$ alongside xP in a standard double-and-add fashion, resulting in the special case of Algorithm 1 in which all digits x_i are 0 or 1. To avoid field inversions, the numerator and the denominator of the function are kept separate; the correctness of the algorithm may be derived from the loop invariant $f_{x',P}(Q) = f/g$, where $x' = \sum_{j=0}^{n-i-1} x_{i+j+1} 2^j$.

Algorithm 1: Miller's algorithm using a NAF

Input: $P \neq \mathcal{O}, Q \neq \lambda P$, two points on an elliptic curve E over a field
$x = \sum_{i=0}^{n} x_i 2^i$ with $x_i \in \{-1, 0, 1\}$
Output: $f_{x,P}(Q)$
$R \leftarrow P$
$f \leftarrow 1$, $g \leftarrow 1$
for $i \leftarrow n-1$ **downto** 0 **do**
  $f \leftarrow f^2 \ell_{R,R}(Q)$
  $R \leftarrow 2R$
  $g \leftarrow g^2 v_R(Q)$
  if $x_i = 1$ **then**
    $f \leftarrow f \ell_{R,P}(Q)$
    $R \leftarrow R + P$
    $g \leftarrow g v_R(Q)$
  if $x_i = -1$ **then**
    $f \leftarrow f \ell_{R,-P}(Q)$
    $R \leftarrow R - P$
    $g \leftarrow g v_R(Q) v_P(Q)$
return f/g

Techniques for speeding up scalar products on elliptic curves may be adapted. For instance, allowing addition-subtraction chains for computing xP, preferably by writing x in non-adjacent form (NAF), in which no two consecutive digits are non-zero, results in Algorithm 1. Notice the additional multiplication by v_P in the case of a digit -1, due to $f_{-1,P} = 1/v_P$, which is often incorrectly left out in the literature; in particular, [17, Algorithm 1] is only correct for addition chains. The common omission may be explained by the use of denominator elimination: If k is even and $G_2 = E(\mathbb{F}_{q^k}) \cap \ker(\pi - q)$, then the x-coordinates of P and Q and thus all values $v_R(Q)$ lie in the subfield $\mathbb{F}_{q^{k/2}}$ and disappear for reduced pairings involving a final exponentiation. In particular, g may be dropped completely.

An observation made in [7] allows to simplify the expression in the case $x_i = -1$ also for odd k. Denote by λ the slope of the line between R and $-P$. Then

$$\ell_{R,-P}(Q) = y(Q) + y(P) - \lambda\big(x(Q) - x(P)\big),$$

which is computed with one multiplication, and f and g are updated with four multiplications

altogether, or with two multiplications if denominator elimination applies. Then

$$\ell'_{R,-P}(Q) := \frac{\ell_{R,-P}(Q)}{v_P(Q)} = \frac{y(Q)+y(P)}{x(Q)-x(P)} - \lambda \tag{2}$$

is obtained without any multiplication since the first term may be precomputed once and for all. So replacing the block in the case of $x_i = -1$ by

$$f \leftarrow f\ell'_{R,-P}(Q), \quad R \leftarrow R - P, \quad g \leftarrow g v_R(Q)$$

reduces the number of multiplications to 2, the same as in the presence of denominator elimination.

Algorithm 2: Boxall *et al.*'s algorithm using a NAF

Input: $P \neq \mathcal{O}, Q \neq \lambda P$, two points on an elliptic curve E over a field
$x = \sum_{i=0}^{n} x_i 2^i$ with $x_i \in \{-1, 0, 1\}$
Output: $f_{x,P}(Q)$
$h \leftarrow n + \#\{x_i | x_i \neq 0\}_{0 \leq i \leq n} - 1$
$\delta \leftarrow (-1)^h$
$g \leftarrow f_{\delta,P}(Q)$, where $f_{1,P} = 1$ and $f_{-1,P} = 1/v_P$
$f \leftarrow 1, R \leftarrow P$
for $i \leftarrow n-1$ **downto** 0 **do**
  if $\delta = 1$ **then**
    $f \leftarrow f^2 \ell_{R,R}(Q)$ // $f_{-2n,P} = (f^2_{n,P}\ell_{[n]P,[n]P})^{-1}$
    $g \leftarrow g^2$
    $R \leftarrow 2R,\ \delta \leftarrow -\delta$
    if $x_i = 1$ **then** // $f_{n+1,P} = (f_{-n,P}f_{-1,P}\ell_{[-n]P,-P})^{-1}$
      $g \leftarrow g\ell'_{-R,-P}(Q)$
      $R \leftarrow R + P,\ \delta \leftarrow -\delta$
    if $x_i = -1$ **then** // $f_{n-1,P} = (f_{n,P}\ell_{[-n]P,P})^{-1}$
      $g \leftarrow g\ell_{-R,P}(Q)$
      $R \leftarrow R - P,\ \delta \leftarrow -\delta$
  else
    $g \leftarrow g^2 \ell_{-R,-R}(Q)$ // $f_{2n,P} = (f^2_{-n,P}\ell_{[-n]P,[-n]P})^{-1}$
    $f \leftarrow f^2$
    $R \leftarrow 2R,\ \delta \leftarrow -\delta$
    if $x_i = 1$ **then** // $f_{-(n+1),P} = (f_{n,P}\ell_{[n]P,P})^{-1}$
      $f \leftarrow f\ell_{R,P}(Q)$
      $R \leftarrow R + P,\ \delta \leftarrow -\delta$
    if $x_i = -1$ **then** // $f_{-(n-1),P} = (f_{n,P}f_{-1,P}\ell_{[n]P,-P})^{-1}$
      $f \leftarrow f\ell'_{R,-P}(Q)$
      $R \leftarrow R - P,\ \delta \leftarrow -\delta$
return f/g

In the same article [7], Boxall *et al.* introduce a variant of the algorithm based on

$$f_{n+m,P} = (f_{-n,P}f_{-m,P}\ell_{-nP,-mP})^{-1},$$

which contains only three factors instead of the four in (1), but requires a swap of the numerator and the denominator for each doubling or adding on the elliptic curve. The algorithm of [7] is given only for addition chains, but can be generalised to addition-subtraction chains, yielding Algorithm 2. The correctness of the algorithm stems from the equations given in commentary and (2), and the loop invariant $f_{\delta x',P}(Q)^\delta = f/g$, where $x' = \sum_{j=0}^{n-i-1} x_{i+j+1} 2^j$. The value of h is the total number of curve doublings, additions and subtractions carried out during the algorithm; δ is $+1$ or -1, respectively, depending on whether the number of curve operations still to be carried out is odd or even. In the end, $\delta = 1$, and f/g is the desired result.

4 Final exponentiation

After the evaluation of a rational function, most pairings require a final powering to obtain a uniquely defined result. It has been suggested that at high levels of security the final exponentiation would become so computationally expensive that the Weil pairing should be preferred to the Tate pairing [16], but this conclusion was quickly contradicted by a finer analysis of the exponentiation step [11]. The rather special form of the exponent $(q^k - 1)/r$ makes the final powering less daunting than it may first appear. This exponent can be decomposed into three parts as follows. Let i be the smallest prime divisor of k.

$$\frac{q^k - 1}{r} = \underbrace{(q^{k/i} - 1) \cdot \frac{q^k - 1}{(q^{k/i} - 1)\Phi_k(q)}}_{\text{easy}} \cdot \underbrace{\frac{\Phi_k(q)}{r}}_{\text{hard}}$$

where Φ_k is the k-th cyclotomic polynomial. The first two factors are sums of powers of q and are easily computed using only a few applications of the Frobenius map $x \mapsto x^q$ and multiplications in G_T, together with an extra division in G_T for the first part. Consequently, the powering by $N = \Phi_k(q)/r$, dubbed the "hard part", is the only truly expensive step of the final exponentiation. It is usually carried out with multi-exponentiation techniques.

Generic exponentiation An algorithm proposed by Avanzi and Mihăilescu in [3] makes particularly intensive use of the Frobenius map, which is very efficient for finite field extensions given by binomials, see Table 4. To compute b^N, it first decomposes N in base q, then each coefficient in base 2^ℓ for a suitably chosen small ℓ, such that $N = \sum_{i=0}^{\lceil \log_q N \rceil} \sum_{j=0}^{\lceil \log_{2^\ell}(q-1) \rceil} n_{ij} q^i 2^{j\ell}$, and $b^N = \prod_j \left(\prod_i b^{n_{ij} q^i} \right)^{2^{j\ell}}$. After precomputing all possible values of $b^{n_{ij}}$ with about 2^ℓ multiplications, for each j the inner product is evaluated in a Horner scheme; altogether, this requires $O((\log N)/\ell)$ Frobenius maps and as many multiplications. The outer product is then computed by a square-and-multiply approach with $O(\log q) = O((\log N)/k)$ multiplications, most of which are squarings. Notice that for $2^\ell \in O(\log N / \log\log N)$ and k growing sufficiently fast with respect to N (which is the case due to the security equivalence of §2), the total complexity becomes $O(\log N / \log\log N)$ operations, which is sublinear in the bit size of N.

As can be seen in Table 4, the Frobenius is not completely negligible when the field extension is realised by a trinomial, so we investigated an alternative approach due to Nogami *et al.* [18], which purportedly requires fewer applications of the Frobenius map. Let ℓ be an integer and $c = \lceil (\log_q N)/\ell \rceil$, and let $t = \lceil \log_2 q \rceil$. The algorithm of [18] creates a binary matrix with ℓ rows and ct columns by first writing N in basis q^c. Each coefficient corresponds to one row and is decomposed into c coefficients in base q, each of which is in turn written with t coefficients in base 2. These form the columns of the matrix, organised into c blocks of t columns each. To compute b^N, first the powers b^{2^i} are precomputed with $t - 1$ multiplications. If the same column occurs

$d \geq 2$ times in $e \leq c$ column blocks, its occurrences can be combined with $d-1$ multiplications and $e-1$ applications of the Frobenius map. Taking into account that there are at most $2^\ell - 1$ different non-zero columns, this step can thus be carried out with at most $ct-1$ multiplications and $(c-1)(2^\ell-1)$ Frobenius maps. Heuristically, a fraction of $1/2^\ell$ of the columns are zero, for which there is nothing to do; so one expects a number of multiplications closer to $ct(1-1/2^\ell)$, a noticeable difference for small values of ℓ. For each row, the combined columns which have a 1 in this row need to be multiplied, with at most $\ell(2^{\ell-1}-1)$ multiplications. Finally, the rows are combined with $\ell-1$ multiplications and $\ell-1$ Frobenius maps. We arrive at a complexity of $(c-1)(2^\ell-1)+\ell-1$ Frobenius maps and $ct+\ell(2^{\ell-1}-1)+\ell+t-3$ multiplications. The latter can be tightened heuristically by multiplying the first term with $1-1/2^\ell$, which turns out to be close to the experimentally observed values. Asymptotically, the number of multiplications is equivalent to $(\log_2 N)/\ell + \ell 2^\ell$. Recall from the security equivalence of §2 that k is of the order of $\log^2 q$, so that $c \in O\left((\log N)^{2/3}/\ell\right)$ and the number of Frobenius maps is in $O\left(2^\ell(\log N)^{2/3}/\ell\right)$. Letting $2^\ell = (\log N)^{1/3}$ yields a sublinear total complexity of $O(\log N/\log\log N)$. The analysis also shows that by preferring smaller values of ℓ, one may reduce the number of Frobenius maps compared to Avanzi and Mihăilescu's algorithm, at the price of more multiplications. Notice that ℓ fixes c, and that the exponentiality in ℓ implies that an optimal value is found easily in practice.

Family-dependant exponentiation Scott *et al.* proposed in [21] an exponentiation technique for polynomial families of parameters $q(x)$ and $r(x)$. The exponent is written first in base $q(x)$, then each coefficient in base x as

$$N(x) = \Phi_k(q(x))/r(x) = \sum_{i=0}^{\lfloor \deg N(x)/\deg q(x) \rfloor} \sum_{j=0}^{\deg q(x)-1} \lambda_{i,j} x^j q(x)^i.$$

To obtain $b^{N(x_0)}$, the values $b^{x_0^j q^i}$ are precomputed with first $\deg q(x)$ exponentiations by x_0, which can be done with $O(\log q)$ multiplications, then about $\deg N(x)$ applications of the Frobenius map. The final result is then obtained by multi-exponentiation with the exponents $\lambda_{i,j}$. The exact complexity of this step depends on the length of an addition sequence passing through all of the $\lambda_{i,j}$. If $\Lambda = \max|\lambda_{i,j}|$, then the best rigorous bound currently available is $\log_2\Lambda + \deg N(x) O(\log\Lambda/\log\log\Lambda)$, where $\deg N(x)$ is the number of potentially different values of $\lambda_{i,j}$, realised, for instance, by [24]. In practice, the coefficients are small, and there are addition sequences with only few additional terms, see Table 12, leading to a heuristic complexity of $O(\deg N(x))$ multiplications. The total complexity then becomes $O(\log q + \deg N(x))$ multiplications and $O(\deg N(x))$ Frobenius maps, where $\deg N(x) \approx \log N/\log x_0$.

5 Implementation

We have implemented the various pairings for the different curves of §2 in the PARI library and linked them into the free number theoretic computer algebra system GP [15]. Our aim was not to provide an optimal *ad hoc* implementation for any one of the curves or pairings, but rather to keep a sufficient level of genericity appropriate for a general purpose system, while still implementing algorithmic optimisations that apply in a broader context. All benchmarks were performed on a Macbook Pro with a 2.5 GHz Core 2 Duo processor, and timings are given in milliseconds (rounded to two significant digits).

5.1 Finite field arithmetic

Arithmetic in $\mathbb{F}_p$ is already available in PARI. The field extensions $\mathbb{F}_{p^k}$ are defined by binomials $X^k - a$ for all curves but E_{11}, E_{19} and E_{25}, for which only trinomials of the form $X^k + X + a$

exist. A defining binomial can be found if and only if all prime factors of k divide $p-1$, and, additionally for $4 \mid k$, if $p \equiv 1 \pmod 4$ [19, Theorem 2], which happens quite often in our context where k has only few prime factors and there is a certain freedom in the choice of p. Fast reduction modulo binomials and trinomials had to be added to PARI. Definition by a binomial is crucial for an efficient pairing implementation, see Table 3, which shows the effect for the ate pairing on E_{18} when artificially switching to a trinomial.

Defining polynomial	unreduced ate	unreduced optimal
$x^{18}+x+6$	490 ms	120 ms
$x^{18}+19$	150 ms	35 ms

Table 3: Timings for ate pairing depending on the finite field defining polynomial

In several places in the literature, it is suggested to build $\mathbb{F}_{p^k}$ by successive extensions of degree dividing k, in particular of degree 2 or 3. It is unclear where the benefits of this strategy lie for multiplication: Virtually the same effect may be reached by using Karatsuba (degree 2) and Toom-Cook (degree 3) for polynomial arithmetic, which moreover speeds up the computations also when k is not a power product of 2 and 3. By keeping a single extension, it is also easier to reach the thresholds for asymptotically fast algorithms. In particular, PARI uses Kronecker substitution to replace the product of polynomials by that of large integers which is outsourced to the GMP library [12]; in experiments, this turned out to be faster than Karatsuba multiplication.

Note that using a binomial to define the field extensions also indirectly speeds up the arithmetic when field elements lie in subfields of $\mathbb{F}_{q^k}$, which happens systematically for the group $G_2 = E[r] \cap \ker(\pi - q)$ in the presence of twists. As an example consider again the curve E_{18}. Let $D \in \mathbb{F}_{p^3}$ be a quadratic and a cubic non-residue in $\mathbb{F}_{p^3}$, which implies that $\mathbb{F}_{p^{18}} = \mathbb{F}_{p^3}[D^{1/6}]$. Then E_{18}: $y^2 = x^3 + b$ admits a sextic twist $E' : y^2 = x^3 + b/D$, explicitly given by $\phi_6 : E'(\mathbb{F}_3) \to E(\mathbb{F}_{18}) : (x, y) \mapsto \left(\sqrt[3]{D}x, \sqrt{D}y\right)$, which yields an isomorphism of $E'(\mathbb{F}_{p^3})[r]$ with G_2. If $\mathbb{F}_{18}$ is defined by a binomial $X^{18} + A$, then $D = A^{1/3} = X^6$, the elements of $\mathbb{F}_{p^3}$ are written as $a_{12}X^{12} + a_6X^6 + a_0$, and an element Q of G_2 is given as $\phi_6(Q') = (a_{14}X^{14} + a_8X^8 + a_2X^2, a_{15}X^{15} + a_9X^9 + a_3X^3)$ with all $a_i \in \mathbb{F}_p$. These sparse polynomials lead to a faster arithmetic, and part of the speed gains for binomial field extensions as shown in Table 3 may be attributed to this implicit handling of subfields.

Explicit towers of finite fields could be useful for realising the Frobenius automorphism of $\mathbb{F}_{p^k}/\mathbb{F}_p$, since the non-trivial automorphism of a quadratic extension is a simple sign flip; while those of cubic extensions require a multiplication in the smaller field anyway. We chose instead to implement the Frobenius, as well as its powers, as linear maps by multiplying with precomputed $k \times k$ matrices. We originally intended to study the use of optimal normal bases of $\mathbb{F}_{p^k}/\mathbb{F}_p$, in which the Frobenius π is realised as a simple permutation of the coordinates. It turns out, however, that again binomial field extensions yield an efficient arithmetic: In their presence, the matrix of π is the product of a diagonal matrix and a permutation, so π and its powers can be computed with k multiplications in $\mathbb{F}_p$ [4, Theorem 3]. In the trinomial case, the Frobenius is computed with close to k^2 multiplications in $\mathbb{F}_p$.

Table 4 summarises the relative costs of the Frobenius π, the multiplication M_2 and the inversion I_2 in $\mathbb{F}_{p^k}$ and the multiplication M_1 and the inversion I_1 in $\mathbb{F}_p$. The effect of a defining trinomial on the cost of π is clearly visible.

	E_9	E_{11}	$E_{12,1}$	$E_{12,2}$	E_{15}	E_{16}	E_{17}	E_{18}	E_{19}	$E_{12,3}$	E_{24}	E_{25}	E_{26}	E_{27}
M_1/μs	0.41	0.36	0.27	0.82	0.64	0.57	0.66	0.64	0.49	2.2	0.74	0.83	0.73	0.64
I_1/M_1	11	11	15	12	13	14	12	12	13	10	13	12	12	13
M_2/M_1	55	90	110	80	120	130	130	140	170	70	250	210	250	240
I_2/M_2	8.0	7.9	8.6	8.1	8.1	8.7	9.2	8.8	8.7	8.1	9.2	9.5	10	10
π/M_2	0.19	0.63	0.19	0.17	0.15	0.15	0.15	0.16	0.95	0.18	0.14	1.2	0.16	0.15

Table 4: Costs of arithmetic in finite prime and extension fields

5.2 Miller loop

Given the cost of inversion in Table 4, we implemented the elliptic curve arithmetic using affine coordinates. Timings for the Miller loop are summarised in Table 5. While mathematically not necessary, the Tate and Weil pairings have also been restricted to the subgroups G_1 and G_2 of eigenvalue 1 and p, respectively, which assures a fairer comparison and incidentally a type 3 pairing in the notation of [10], see also [8]. For even embedding degree, we applied denominator elimination. The first row uses a double-and-add approach, the second one a signed NAF with Algorithm 1. The variant of Algorithm 2 is only of interest when k is odd; we give its timings with a double-and-add chain and a NAF in the third and fourth rows, respectively. It makes an impressive difference.

Generically, one expects a NAF to save about 11% of the number of operations. For our curves, the effect is often much less. This can be explained by the sparsity of the integer r derived from a curve family, which is thus closer to non-adjacent form than a random integer. For instance, the binary decomposition of r for $E_{12,1}$ has only 87 entries 1 out of 256, a density that would be expected in a NAF of a random number. The NAF has 37 entries 1 and -1 each. Also counting the squarings, the gain in the number of operations is less than 4%. One could reduce the number of multiplications even further by combining with a sliding window technique; since the number of squarings is unchanged, the effect will be more and more marginal with an increasing window size.

At these high security levels and consequently big values of k, the ate pairing e_{A} is clearly in general slower than the Tate pairing; the smaller multiplier is more than offset by the need to work over the extension field. The only exception is E_{24} with a particularly small trace ($t(x)$ of degree 1 for $r(x)$ of degree 8). In fact, the ate pairing for this curve is already optimal. As can be expected, the twisted ate pairing $e_{\mathrm{A}'}$ is even less efficient except for small values of k combined with a high degree twist: The power of the trace $T^e = T^{k/d}$ quickly exceeds r itself (in the table, we computed with T^e; one could reduce modulo r and arrive at the same timings as the Tate pairing, but may then as well stick with the original). The optimal versions indeed keep their promises. Due to the overhead of computing several functions, the total running time is not reduced by a factor of $\varphi(k)$, but the optimal ate pairing is generally faster than the Tate pairing. Twisted pairings are asymptotically slower, but interesting for the medium values of $k = 9$, 12 or 16 which admit a twist of (relatively) high degree 3, 6 or 4.

The Weil pairing with its two function evaluations could be expected to behave like a Tate followed by an ate pairing; due to the different loop lengths, the part $f_{r,Q}(P)$, however, has a complexity closer to $\deg r(x)/\deg t(x)$ times that of ate, as can be roughly verified in the table. As already stated in the literature, the enormous overhead of the Weil pairing is not offset by saving the final exponentiation, see §5.3.

At higher security levels, odd values of k lead to a bigger $\varphi(k)$ and thus a higher gain in the optimal pairing; together with the Miller loop improvement of [7], odd and in particular prime values of k become attractive. Notice that E_{11}, E_{19} and E_{25} are heavily penalised by the trinomial field extension. Indeed, odd or, worse, prime values of k make the divisibility conditions for the existence of a binomial extension harder to satisfy. Moreover, the degree of

Curve	e_{T}	$e_{\mathrm{A}'}$	$e_{\mathrm{O}'}$	e_{A}	e_{O}	e_{W}	$\deg q$	$\deg r$	$\deg t$	$\varphi(k)$
E_9	31	64	15	50	14	100	8	6	4	6
	29	59	15	45	14	97				
	21	46	11	46	12	85				
	20	42	11	42	12	80				
E_{11}	43	–	–	110	20	230	24	20	12	10
	39	–	–	100	20	200	24	20	12	10
	31	–	–	100	20	200				
	28	–	–	96	19	180				
$E_{12,1}$	14	13	7	18	9	70	4	4	2	4
	14	13	7	18	9	67				
$E_{12,2}$	93	91	53	110	54	420	4	4	2	4
	91	91	52	110	54	410				
E_{15}	130	520	73	270	44	460	12	8	6	8
	130	480	71	240	41	440				
	90	360	53	240	41	370				
	88	330	50	220	37	360				
E_{16}	64	170	35	150	230	360	10	8	5	8
	62	160	34	140	220	340				
E_{17}	160	–	–	54	54	930	38	32	2	16
	140	–	–	50	50	820				
	110	–	–	51	51	830				
	100	–	–	47	47	740				
E_{18}	78	160	39	160	35	400	8	6	4	6
	75	150	36	150	35	380				
E_{19}	130	–	–	350	40	760	40	36	20	18
	120	–	–	320	35	680				
	93	–	–	340	40	670				
	88	–	–	300	34	610				
$E_{12,3}$	410	400	240	450	220	1800	4	4	2	4
	400	400	240	450	230	1700				
E_{24}	190	88	88	55	55	960	10	8	1	8
	180	85	85	56	56	900				
E_{25}	450	–	–	1400	120	2700	52	40	26	20
	410	–	–	1300	130	2400				
	310	–	–	1400	130	2400				
	300	–	–	1200	120	2200				
E_{26}	210	–	–	660	99	1600	28	24	14	12
	190	–	–	610	90	1400				
E_{27}	370	1900	190	570	54	1300	20	18	10	18
	350	1800	180	510	53	1300				
	260	1300	130	530	52	1100				
	250	1200	130	450	47	1100				

Table 5: Timings of the function evaluation in milliseconds

$p(x)$ also grows with $\varphi(k)$, so that the polynomial represents fewer numbers in the desired range and leaves less choice for p or a value of x_0 with low Hamming weight. Even if a binomial field

extension exists, odd values of k that are not divisible by 3 (in particular, prime k again) suffer from a lack of twists and thus a less efficient field arithmetic as discussed in §5.1.

5.3 Final exponentiation

Timings for the final exponentiation are compiled in Table 6.

Curve	$\varphi(k)$	Naive	Hard naive	AM04	NMKM08	SBCPK09
E_9	6	56	36	15	15	8
E_{11}	10	80	78	28	24	21
$E_{12,1}$	4	58	17	8	8	4
$E_{12,2}$	4	380	100	41	37	26
E_{15}	8	490	250	85	110	50
E_{16}	8	420	200	66	80	44
E_{17}	16	580	550	180	200	110
E_{18}	6	680	210	78	83	49
E_{19}	18	460	460	150	110	83
$E_{12,3}$	4	1800	470	170	170	120
E_{24}	8	2000	640	150	200	97
E_{25}	20	2600	2100	700	470	320
E_{26}	12	2300	1000	240	270	170
E_{27}	18	2100	1400	290	310	130

Table 6: Final exponentiation times in milliseconds

The first column corresponds to a direct exponentiation by $(q^k - 1)/r$ via the sliding window algorithm built into PARI. The second column does so for the hard part, while computing the easy one using Frobenius maps. The next two columns relate the implementation of the hard parts following [3] and [18]. At low security level, the differences between these two algorithms are minimal. For the medium and high security range, our implementation confirms the claim of [18]: Their algorithm becomes faster when Frobenius maps are more expensive, as for the three curves E_{11}, E_{19} and E_{25}. The $k = 12$ curves stand out: The low value of $\varphi(k)$ makes the final exponentiation much easier with these two algorithms that rely on an expansion in base q.

While the theoretical analysis of §4 is not conclusive, the experiments are unequivocal: The algorithm of [21] (which we used, as explained in the article, to potentially compute a small power of the true pairing if the coefficients of the polynomial contain denominators) is clearly the fastest one for curves obtained from polynomial families.

6 Overall timings and conclusion

For each of our reference curves, Table 7 summarises the timings obtained for the fastest pairing.

Optimal pairings are indeed optimal for higher security levels. Their unreduced version benefits from high values of $\varphi(k)$, as can be seen by comparing E_{24} and E_{27}. However, part of this advantage is offset by the lack of denominator elimination for odd k, although Boxall *et al.*'s variant almost closes the gap again. Moreover, the higher cost for the final exponentiation more than compensates the gain in the Miller loop. The decision which pairing to take then also depends on the concrete cryptographic protocol: Not all of them require reduced pairings throughout their execution. For instance, verification protocols such as [6] make do with testing

Security	Curve	$\varphi(k)$	Pairing	Unreduced	Final exp	Reduced
128 bit	E_9	6	$e_{\mathrm{O}'}$	11	8	19
	E_{11}	10	e_{O}	19	21	40
	$E_{12,1}$	4	$e_{\mathrm{O}'}$	7	4	11
192 bit	$E_{12,2}$	4	$e_{\mathrm{O}'}$	52	26	78
	E_{15}	8	e_{O}	37	50	87
	E_{16}	8	$e_{\mathrm{O}'}$	34	44	78
	E_{17}	16	$e_{\mathrm{A}} = e_{\mathrm{O}}$	47	110	157
	E_{18}	6	e_{O}	35	49	84
	E_{19}	18	e_{O}	34	83	120
256 bit	$E_{12,3}$	4	e_{O}	220	120	340
	E_{24}	8	$e_{\mathrm{A}} = e_{\mathrm{O}}$	55	97	150
	E_{25}	20	e_{O}	120	320	440
	E_{26}	12	e_{O}	90	170	260
	E_{27}	18	e_{O}	47	130	180

Table 7: Timings of the fastest reduced pairing variants

equality of products of several pairings. All of these may then be computed unreduced, and only a final quotient of products needs to be raised to the power, which makes this exponentiation negliglible.

For a reduced pairing at lower security levels, Barreto–Naehrig curves with $k = 12$ remain unbeaten, profiting from an exceptionally fast final exponentiation.

At 192 bit security, Barreto–Naehrig curves need to work with a larger than optimal size of the underlying elliptic curve, but still provide the fastest pairings. An equivalent performance, however, may be reached for $k = 16$ with curve E_{16}. The suboptimal $\rho = 5/4$ notwithstanding, this curve is of size 501 bits instead of 663 bits for the Barreto–Naehrig curve, resulting in less bandwidth for exchanging curve points. Thus our study shows that E_{16} is preferable at medium security level.

At the highest AES equivalent of 256 bit, Barreto–Naehrig curves are no longer competitive speed-wise. Here the curve E_{24} stands out. Although gaining only a factor of 8 in the Miller loop length, it profits from a very fast final exponentiation, while even the unreduced variant remains comparable to the closest competitor E_{27}.

As becomes clear from this study, extension fields $\mathbb{F}_{p^k}$ that do not allow a binomial as a defining polynomial are to be banned, see E_{11}, E_{19} and E_{25}.

Whether odd or even embedding degrees are preferable remains undecided. Our results seem to indicate that odd degrees are slightly slower. This can be explained by their higher probability of requiring a trinomial field extension, sparser families and the lack of twists as explained at the end of §5.2. Often, the gain odd and, in particular, prime embedding degrees provide through larger values of $\varphi(k)$ for the optimal pairings is more than offset by an expensive final exponentiation, as is well illustrated by E_{17}. In protocols that work with mostly unreduced pairings, however, Boxall *et al.*'s variant of the Miller loop makes odd embedding degrees competitive, see E_{15} and E_{27}.

A definite conclusion is made difficult by the lack of choice for any given security level: Some families are so sparse that they contain no curves of prime cardinality in the desired range or, if they do, no curves allowing to work with extension fields defined by binomials. Even if suitable curves exist, the sparsity of a family may have a big impact on the efficiency of the Miller loop. Notice that the loop for E_{27} is more than twice shorter than that of E_{24}. Nevertheless, the

unreduced pairing is computed in almost the same time. This can be explained by the Hamming weight of the multiplier: The family of E_{27} is instantiated with x_0 of weight 13, that of E_{24} with x_0 of weight 7. So the search for new curve families remains a research topic of interest, not only for families with optimal ρ, as witnessed by the good performance of E_{15} despite its very bad $\rho = 3/2$.

A Curve parameters

Tables 8 to 10 give the exact parameters for the curves we studied for different security levels. The following notations are used: $p(x)$, $r(x)$ and $t(x)$ are the polynomials representing the cardinality of the finite prime field $\mathbb{F}_q$, a (large) prime factor of the curve cardinality and the trace of Frobenius, respectively; x_0 is the numeric value of the variable x; (a, b) gives the equation of the curve $y^2 = x^3 + ax + b$; F is the irreducible polynomial defining $\mathbb{F}_{p^k}$. For $k = 16$ or 17, no prime values $r(x_0)$ exist in the desired range. We thus admit a small cofactor and let r_0 denote the actual large prime factor of $r(x_0)$.

Table 11 provides the short lattice vectors yielding our optimal ate pairings e_{O} and optimal twisted ate pairings $e_{\mathrm{O}'}$, see §2.2.

Table 12 records the addition sequences used in the final exponentiation of [21]. To remove denominators, the power s of the original pairing is computed; n is the number of (not necessarily distinct) non-zero coefficients λ_{ij}, see §4. The underlined terms are those that are added to the sequence. As can be seen, there is in general a very small number of very small distinct coefficients, and only a tiny number of terms, if any, needs to be added.

E_9	$p(x) = (x^8 - x^7 + x^6 - x^5 - 2x^4 - x^3 + x^2 + 2x + 1)/3$ $r(x) = (x^6 - x^3 + 1)/3$ $t(x) = -x^4 + x + 1$ $(a, b) = (0, 7) \quad F(X) = X^9 + 3 \qquad x_0 = 43980465324080$
E_{11}	$p(x) = (x^{24} - x^{23} + x^{22} - x^{13} + 4x^{12} - x^{11} + x^2 - x + 1)/3$ $r(x) = x^{20} + x^{19} - x^{17} - x^{16} + x^{14} + x^{13} - x^{11} - x^{10} - x^9 + x^7 + x^6 - x^4 - x^3 + x + 1$ $t(x) = x^{12} + 1$ $(a, b) = (0, 4) \quad F(X) = X^{11} + X + 11 \quad x_0 = 11210$
$E_{12,1}$	$p(x) = 36x^4 + 36x^3 + 24x^2 + 6x + 1$ $r(x) = 36x^4 + 36x^3 + 18x^2 + 6x + 1$ $t(x) = 6x^2 + 1$ $(a, b) = (0, 5) \quad F(X) = X^{12} + 5 \qquad x_0 = 6917529027641094616$

Table 8: Curves for security level 128 bit

E_{15}	$p(x) = (x^{12} - x^{11} + x^{10} - x^7 - 2x^6 - x^5 + x^2 + 2x + 1)/3$ $r(x) = x^8 + x^7 - x^5 - x^4 - x^3 + x + 1$ $t(x) = -x^6 + x + 1$ $(a, b) = (0, 13) \quad F(X) = X^{15} + 13 \qquad x_0 = 271533021386417$
E_{16}	$p(x) = (x^{10} + 2x^9 + 5x^8 + 48x^6 + 152x^5 + 240x^4 + 625x^2 + 2398x + 3125)/980$ $r(x) = (x^8 + 48x^4 + 625) \qquad r_0 = r(x_0)/20641250$ $t(x) = (2x^5 + 41x + 35)/35$ $(a, b) = (1, 0) \quad F(X) = X^{16} + 2 \qquad x_0 = 2251799888961585$
E_{17}	$p(x) = (x^{38} + 2x^{36} + x^{34} + x^4 - 2x^2 + 1)/4$ $r(x) = x^{32} - x^{30} + x^{28} - x^{26} + x^{24} - x^{22} + x^{20} - x^{18} + x^{16} - x^{14} + x^{12} - x^{10}$ $+x^8 - x^6 + x^4 - x^2 + 1 \qquad r_0 = r(x_0)/12071636373225929$ $t(x) = -x^2 + 1$ $(a, b) = (13, 0) \quad F(X) = X^{17} + 2 \qquad x_0 = 12681$
E_{18}	$p(x) = (x^8 + 5x^7 + 7x^6 + 37x^5 + 188x^4 + 259x^3 + 343x^2 + 1763x + 2401)/21$ $r(x) = (x^6 + 37x^3 + 343)/343$ $t(x) = (x^4 + 16x + 7)/7$ $(a, b) = (0, 19) \quad F(X) = X^{18} + 19 \qquad x_0 = 48422703193491756920$
E_{19}	$p(x) = (x^{40} - x^{39} + x^{38} - x^{21} - 2x^{20} - x^{19} + x^2 + 2x + 1)/3$ $r(x) = x^{36} + x^{35} - x^{33} - x^{32} + x^{30} + x^{29} - x^{27} - x^{26} + x^{24} + x^{23} - x^{21} - x^{20} + x^{18}$ $-x^{16} - x^{15} + x^{13} + x^{12} - x^{10} - x^9 + x^7 + x^6 - x^4 - x^3 + x + 1$ $t(x) = -x^{20} + x + 1$ $(a, b) = (0, 9) \quad F(X) = X^{19} + X + 23 \quad x_0 = 1274$
$E_{12,2}$	$p(x) = 36x^4 + 36x^3 + 24x^2 + 6x + 1$ $r(x) = 36x^4 + 36x^3 + 18x^2 + 6x + 1$ $t(x) = 6x^2 + 1$ $(a, b) = (0, 13) \quad F(X) = X^{12} + 5$ $x_0 = 29230032746618058364073696654325660393118650866996$

Table 9: Curves for security level 192 bit

E_{24}	$p(x) = (x^{10} - 2x^9 + x^8 - x^6 + 2x^5 - x^4 + x^2 + x + 1)/3$ $r(x) = x^8 - x^4 + 1$ $t(x) = x + 1$ $(a, b) = (0, 1) \quad F(X) = X^{24} + 19 \qquad x_0 = 18446744073709602433$
E_{25}	$p(x) = (x^{52} - x^{51} + x^{50} - x^{27} - 2x^{26} - x^{25} + x^2 + 2x + 1)/3$ $r(x) = x^{40} + x^{35} - x^{25} - x^{20} - x^{15} + x^5 + 1$ $t(x) = -x^{26} + x + 1$ $(a, b) = (0, 31) \quad F(X) = X^{25} + X + 19 \quad x_0 = 6995$
E_{26}	$p(x) = (x^{28} + x^{27} + x^{26} - x^{15} + 2x^{14} - x^{13} + x^2 - 2x + 1)/3$ $r(x) = x^{24} + x^{23} - x^{21} - x^{20} + x^{18} + x^{17} - x^{15} - x^{14} + x^{12} - x^{10} - x^9 + x^7 + x^6 - x^4 - x^3 + x + 1$ $t(x) = x^{14} - x + 1$ $(a, b) = (0, 12) \quad F(X) = X^{26} + 4 \qquad x_0 = 2685463$
E_{27}	$p(x) = (x^{20} - x^{19} + x^{18} - x^{11} - 2x^{10} - x^9 + x^2 + 2x + 1)/3$ $r(x) = (x^{18} - x^9 + 1)/3$ $t(x) = -x^{10} + x + 1$ $(a, b) = (0, 9) \quad F(X) = X^{27} + 3 \qquad x_0 = 374298113$
$E_{12,3}$	$p(x) = 36x^4 + 36x^3 + 24x^2 + 6x + 1$ $r(x) = 36x^4 + 36x^3 + 18x^2 + 6x + 1$ $t(x) = 6x^2 + 1$ $(a, b) = (0, 7) \quad F(X) = X^{12} + 2$ $x_0 = 93449432821539816104782199644917905013868322853103558685183070342222113002 9\backslash$ 030240635045913079014

Table 10: Curves for security level 256 bit

Curve	Pairing	Vector
E_9	e_{O}	$\left(\frac{1}{3}(x-2), \frac{1}{3}(x+1), \frac{1}{3}(x+1), -\frac{1}{3}(x+1), -\frac{1}{3}(x-2), -\frac{1}{3}(x-2)\right)$
	$e_{\mathrm{O}'}$	$\left(\frac{1}{3}(x^3+1), \frac{1}{3}(2-x^3)\right)$
E_{11}	e_{O}	$\left(x^2, -x, 1, 0, 0, 0, 0, 0, 0, 0\right)$
$E_{12,i}$	e_{O}	$(6x+2, 1, -1, 1)$
	$e_{\mathrm{O}'}$	$\left(2x+1, 6x^2+2x\right)$
E_{15}	e_{O}	$(1, 0, 0, 0, x, 0, 0, 0)$
	$e_{\mathrm{O}'}$	$\left(x^3+x^2-1, x^4+x^3-x-1\right)$
E_{16}	e_{O}	$((2x-15)/35, -(11x-30)/35, -(2x-1)/7, (x+10)/35,$ $(2x+5)/5, (8x+10)/35, (2x+6)/7, (17x+30)/35)), x = 25 \bmod 70$
	$e_{\mathrm{O}'}$	$\left(49x^4/625, 7+168x^4/625\right)$
E_{17}	e_{O}	$\left(x^2, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\right)$
E_{18}	e_{O}	$(1, 3x/7, 3x/7+1, 0, -2x/7, -2x/7-1)$
	$e_{\mathrm{O}'}$	$\left(18(x/7)^3+1, -(x/7)^3\right)$
E_{19}	e_{O}	$\left(x^2, -x, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\right)$
E_{25}	e_{O}	$\left(x^2, -x, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\right)$
E_{26}	e_{O}	$\left(x^2, x, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0\right)$
E_{27}	e_{O}	$(x, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0)$
	$e_{\mathrm{O}'}$	$\left(x^9, 1\right)$

Table 11: Optimal pairings

Curve	s	n	Addition sequence
E_9	1	6	[1, 2, 3]
E_{11}	3	10	[1, 2, 3, 4, 5, 6]
$E_{12,i}$	1	4	[1, 2, 3, 6, 12, 18, 30, 36]
E_{15}	3	8	[1, 2, 3, 4, 5, 6]
E_{16}	857500	8	[1, 2, 4, 6, 10, 11, 15, 20, 22, 25, 29, 30, 40, 50, 54, 55, 75, 100, 125, 145, 220, 250, 272, 278, 300, 440, 585, 625, 875, 900, 950, 1025, 1100, 1172, 1226, 1280, 1372, 1390, 1750, 1779, 2290, 2780, 2925, 3000, 3300, 4250, 4375, 4704, 4750, 4850, 5125, 9700, 13000, 13250, 15000]
E_{17}	4	16	[1, 2, 3, 5]
E_{18}	1029	6	[1, 2, 3, 4, 5, 7, 14, 15, 21, 25, 35, 49, 54, 61, 62, 70, 87, 98, 112, 131, 224, 245, 249, 273, 319, 343, 350, 364, 434, 450, 504, 581, 609, 784, 931, 1057, 1407, 1715, 1911, 2842, 4753, 4802, 6517]
E_{19}	3	18	[1, 2, 3, 4]
E_{24}	3	8	[1, 2, 3]
E_{25}	3	20	[1, 2, 3]
E_{26}	3	12	[1, 2, 3, 4]
E_{27}	3	18	[1, 2, 3]

Table 12: Addition sequences

Acknowledgement. This research was partially funded by ERC Starting Grant ANTICS 278537.

References

[1] Diego F. Aranha, Laura Fuentes-Castañeda, Edward Knapp, Alfred Menezes, and Francisco Rodríguez-Henríquez. Implementing pairings at the 192-bit security level. In Michel Abdalla and Tanja Lange, editors, *Pairing-Based Cryptography — Pairing 2012*, volume 7708 of *Lecture Notes in Computer Science*, pages 177–195, Berlin, 2013. Springer-Verlag.

[2] Diego F. Aranha, Koray Karabina, Patrick Longa, Catherine H. Gebotys, and Julio López. Faster explicit formulas for computing pairings over ordinary curves. In Kenneth G. Paterson, editor, *Advances in Cryptology — EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 48–68, Berlin, 2011. Springer-Verlag.

[3] Roberto Maria Avanzi and Preda Mihăilescu. Generic efficient arithmetic algorithms for PAFFs (processor adequate finite fields) and related algebraic structures (extended abstract). In Mitsuro Matsui and Robert Zuccherato, editors, *Selected Areas in Cryptography — SAC 2003*, volume 3006 of *Lecture Notes in Computer Science*, pages 320–334, Berlin, 2004. Springer-Verlag.

[4] Daniel V. Bailey and Christof Paar. Efficient arithmetic in finite field extensions with applications in elliptic curve cryptography. *Journal of Cryptology*, 14(3):153–176, 2001.

[5] Razvan Barbulescu, Pierrick Gaudry, Antoine Joux, and Emmanuel Thomé. A heuristic quasi-polynomial algorithm for discrete logarithm in finite fields of small characteristic. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology — EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 1–16, Berlin, 2014. Springer-Verlag.

[6] Olivier Blazy, Georg Fuchsbauer, Malika Izabachène, Amandine Jambert, Hervé Sibert, and Damien Vergnaud. Batch Groth-Sahai. In Jianying Zhou and Moti Yung, editors, *Applied Cryptography and Network Security — ACNS 2010*, volume 6123 of *Lecture Notes in Computer Science*, pages 218–235, Berlin, 2010. Springer-Verlag.

[7] John Boxall, Nadia El Mrabet, Fabien Laguillaumie, and Duc-Phong Le. A variant of Miller's formula and algorithm. In Marc Joye, Atsuko Miyaji, and Akira Otsuka, editors, *Pairing-Based Cryptography — Pairing 2010*, volume 6487 of *Lecture Notes in Computer Science*, pages 417–434, Berlin, 2010. Springer-Verlag.

[8] Sanjit Chatterjee and Alfred Menezes. On cryptographic protocols employing asymmetric pairings – the role of ψ revisited. *Discrete Applied Mathematics*, 159:1311–1322, 2011.

[9] David Freemann, Michael Scott, and Edlyn Teske. A taxonomy of pairing-friendly elliptic curves. *Journal of Cryptology*, 23(2):224–280, 2010.

[10] Steven D. Galbraith, Kenneth G. Paterson, and Nigel P. Smart. Pairings for cryptographers. *Discrete Applied Mathematics*, 156(16):3113–3121, 2008.

[11] R. Granger, D. Page, and N. P. Smart. High security pairing-based cryptography revisited. In Florian Hess, Sebastian Pauli, and Michael Pohst, editors, *Algorithmic Number Theory — ANTS-VII*, volume 4076 of *Lecture Notes in Computer Science*, pages 480–494, Berlin, 2006. Springer-Verlag.

[12] Torbjörn Granlund et al. `gmp` — GNU multiprecision library. Version 5.0.3, `http://gmplib.org/`.

[13] Florian Hess. Pairing lattices. In S. D. Galbraith and K. Paterson, editors, *Pairing-Based Cryptography — Pairing 2008*, volume 5209 of *Lecture Notes in Computer Science*, pages 18–38, Berlin, 2008. Springer-Verlag.

[14] Florian Hess, Nigel P. Smart, and Frederik Vercauteren. The eta pairing revisited. *IEEE Transactions on Information Theory*, 52(10):4595–4602, 2006.

[15] Karim Belabas and the PARI Group, Bordeaux. *PARI/GP*. Version 2.5.0, `http://pari.math.u-bordeaux.fr/`.

[16] Neal Koblitz and Alfred Menezes. Pairing-based cryptography at high security levels. In Nigel P. Smart, editor, *Cryptography and Coding — 10th IMA International Conference*, volume 3796 of *Lecture Notes in Computer Science*, pages 13–36, Berlin, 2005. Springer-Verlag.

[17] Victor S. Miller. The Weil pairing, and its efficient calculation. *Journal of Cryptology*, 17:235–261, 2004.

[18] Yasuyuki Nogami, Hidehiro Kato, Kenta Nekado, and Yoshitaka Morikawa. Efficient exponentiation in extensions of finite fields without fast Frobenius mappings. *ETRI Journal*, 30(6):818–825, 2008.

[19] D. Panario and D. Thomson. Efficient pth root computations in finite fields of characteristic p. *Designs, Codes and Cryptography*, 50(3):351–358, 2009.

[20] Geovandro C.C.F. Pereira, Marcos A. Simplício Jr., Michael Naehrig, and Paulo S.L.M. Barreto. A family of implementation-friendly BN elliptic curves. *Journal of Systems and Software*, 84(8):1319–1326, 2011.

[21] Michael Scott, Naomi Benger, Manuel Charlemagne, Luis J. Dominguez Perez, and Ezekiel J. Kachisa. On the final exponentiation for calculating pairings on ordinary elliptic curves. In Hovav Shacham and Brent Waters, editors, *Pairing-Based Cryptography — Pairing 2009*, volume 5671 of *Lecture Notes in Computer Science*, pages 78–88, Berlin, 2009. Springer-Verlag.

[22] Nigel Smart et al. ECRYPT II yearly report on algorithms and keysizes (2009-2010). Technical Report D.SPA.13, European Network of Excellence in Cryptology II, March 2010. `http://www.ecrypt.eu.org/documents/D.SPA.13.pdf`.

[23] Frederic Vercauteren. Optimal pairings. *IEEE Transactions on Information Theory*, 56(1):455–461, 2010.

[24] Andrew Chi-Chih Yao. On the evaluation of powers. *SIAM Journal on Computing*, 5(1):100–103, March 1976.